UCLA

OFFICE OF ACADEMIC COMPUTING

CCN/NSW SERVICE SUMMARY

FINAL TECHNICAL REPORT

4/9/80–9/30/82

CCN/TR32

Neil Ludlam
Denis De La Roca

UNIVERSITY OF CALIFORNIA, LOS ANGELES

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: OAC/TR32	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): CCN/NSW Service Summary	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report 4/9/80 - 9/30/82
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Neil Ludlam & Denis DeLaRoca	8. CONTRACT OR GRANT NUMBER(s): MDA 903-80-C-0231
9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of California, Office of Academic Computing, Los Angeles, CA 90024	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Prog. Elmt: 62708E; Prog. Code: OT10; ARPA Order No. 2543/12
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209, Attn: Program Management Office	12. REPORT DATE: 10/1/82
	13. NUMBER OF PAGES: 110
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

National Software Works, ARPANET, CCN, File types, Workspace Command interpretor, Interactive services, Batch services, file creation, file management, editing, Library files FORTRAN, PL/I, Assembler, GPSS, SPSS, ECAP, EISPACK, GIM-II, AP-1, B520AS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report sumarizes technical development at UCLA/OAC relating to the Nation Software Works during the period from April 1980 through September 1982. Specifically, it ennumerates the user services that have been installed in NSW, and describes their user interfaces.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102 LF 014 6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UCLA Office of Academic Computing
5628 Math Sciences
University of California C0012
Los Angeles, California 90024

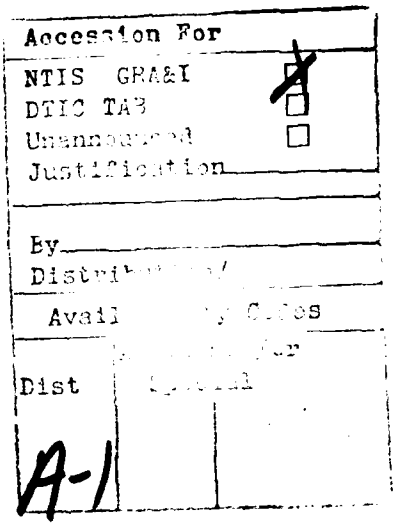
FINAL TECHNICAL REPORT

DARPA Contract
MDA 903-80-C-0231:

ARPANET COMPUTER SERVICES IN SUPPORT OF THE NATIONAL SOFTWARE WORKS

April 9, 1980 - September 30, 1982

William B. Kehl, Principal Investigator
(213)/825-7511



This work was primarily sponsored by the
Defense Advanced Research Projects
Agency under ARPA Order no. 2543/12
Contract no. MDA 903-80-C-0231

Additional support was provided by the
Air Force Systems Command,
Rome Air Development Center,
under Contract no. F30602-82-C-0109

CCN/NSW SERVICE SUMMARY

by
Neil Ludlam
Denis De La Roca

October 1, 1982

Document TR-32

UCLA Office of Academic Computing
5628 Math Sciences
University of California C0012
Los Angeles, California 90024

CONTENTS

LIST OF FIGURES

Figure *page*

Section 1

INTRODUCTION

1.1 WHAT IS CCN?

The National Software Works (NSW) tool-bearing host named "CCN" is an IBM computing system at the UCLA Office of Academic Computing. The name "CCN" dates back to a time when that office was known as the UCLA "Campus Computing Network."

The CCN System consists, at this writing, of a 12-megabyte IBM 3033, one of the larger implementations of the IBM System/370 architecture. It operates under the MVS operating system, and supports both batch computing and interactive access under the IBM Time-Sharing Option (TSO). Both modes of access are available to the NSW user.

1.2 ABOUT THIS MANUAL

This manual describes the NSW services that have been mounted on the CCN host, under the NSW "Development System" as it exists at the time of writing. These services are not necessarily available to NSW users; availability is determined by the NSW administrators.

This manual is intended to be used in conjunction with the NSW User Reference Manual, [26]. It does not purport to describe the National Software Works itself, but only the specific services provided by the CCN host.

Throughout the manual we refer to CCN services by the simple names by which they are known locally. In the larger context of the NSW resource catalogue, these names are not usually sufficient. In most cases, the local name is included literally within the NSW-wide name, so that, for instance, the elliptical form

```
USE ...<simple name>
```

can be used to access a CCN service.

This manual is a major update of several previous documents that describe individual CCN services. It draws from, and supersedes, [27, 28, 29, 30, 31, 32, and 33].

Section 2

HOW TO USE THE CCN NSW SERVICES

2.1 USING CCN BATCH SERVICES

A batch service never does any direct communication with a user. Rather, you converse with components of the NSW Works Manager, to specify interactively the parameters needed to define a "job" for the service, to query that job's status, and to be notified that the job has completed. After the initial job specification is complete, you may end your NSW session if desired, and the job will continue toward its completion. When, in any NSW session, you receive notification that the job has completed, you can examine any output files that the service produced to determine whether the job was successful.

These characteristics of batch services are of interest to the user:

1. In addition to the various files that you may specify for the service's use, the job will produce a "standard system output" file (sometimes abbreviated "SYSOUT"), which is the local operating system's communication to the user of what it did to execute the job, and of what the results seemed to be from its point of view. Ideally, you should not have to be concerned with this file unless you believe that the job was executed improperly. In that case, you should employ the DISPLAY service (not the Front End DISPLAY command) to examine the file. This will reveal much material that is meaningless to those not familiar with IBM systems; however, there are occasional error messages in English.

   The first time you need to examine a SYSOUT file, you should probably simply list the whole file. After examining several such files, you will be able to use the "find" subcommand to pick out relevant information more quickly.

2. All batch jobs must be allocated certain quantities of system resources before they will be executed. For most services, one of these, the number of CPU seconds, varies drastically with the specific job, and must usually be supplied by the user. If a job exceeds this estimate, it will be aborted at that point. On the other hand, if the time is overestimated, the job may not be selected for execution as quickly as it could be. Estimating this quantity is an abstruse art; therefore, each batch service

provides the user with a default value that is judged to be sufficient in "average" cases. Users are advised to accept this default initially, and to note the feedback information listed in the job's SYSOUT file to refine guesses. To locate this feedback, one should use DISPLAY to examine the SYSOUT file and search for a string of the form:

JOB TOTALS (ALL STEPS)

and examine the next few lines. Notice that the CPU time is given in seconds to two decimal places. Most services will ask you to supply an integer representing CPU seconds. Some CPU-intensive services may ask for CPU minutes.

2.2 USING CCN INTERACTIVE SERVICES

The following points should be borne in mind when talking to any interactive process on an IBM host:

1. Communication is half-duplex between the NSW Front End and the IBM process. The terminal remains in "full-duplex" mode, but only because echoing is done by the Front End.

2. Communication is always by entire lines. Nothing reaches the IBM process until you type a carriage return or control-C. Features like command recognition and completion via "escape" are not available.

3. IBM systems use the EBCDIC character code instead of ASCII. Normally, this is transparent to the NSW user, except that many of the "control" characters of an ASCII keyboard are filtered out before reaching the IBM process. However, because the line being typed is being accumulated by the NSW Front End [26], the Front-End control-character conventions for dealing with "line-buffer" hosts are in effect.

4. "Backspace" (control-H) can be accepted by an IBM service. In most cases, it will be interpreted as a character-delete, but some services may process it as data.

5. The IBM "interrupt-process" character is "control-C", entered when there is no incomplete and non-empty input line (otherwise it becomes a line delete). This will also abort any terminal output queued at the service's host (but not any in the network or queued at the Front End).

6. At this writing, CCN does not support recovering and resuming an interactive NSW service session after an interruption by a harware or software system crash. In such a case, your only option will be to restart the service session from the beginning. Some services, such as EDIT, could lose significant work in the event of a crash; however, most CCN interactive services do not accumulate large amounts of data for delivery at the end of the session.

7. At this writing, the CCN encapsulator does not maintain a "Local Name Dictionary" (LND). In general, no relationship between a workspace file and the NSW file of which it is a copy is known to the encapsulator. Specific services may remember such a relationship for specific files -- for instance, the EDIT service is able to deliver its output back to its input file without being told the name again -- but no general mechanism exits.

2.2.1 Talking With The Encapsulator

Interactive services execute in the foreground, under an "Encapsulator". The conversation that collects service parameters from the user is done by the Encapsulator, after the terminal is connected to the IBM host. When all needed parameters are collected, the terminal is turned over to the service itself.

Your conversation with the Encapsulator is limited to answering questions in response to Encapsulator queries. As a matter of convention, there are several valid responses to an Encapsulator query:

1. A naked carriage return: When you are being queried for one of several options, this response tells the Encapsulator to use the default.

2. A question mark: This requests clarifying information from the Encapsulator. If any such information is available, it will be printed, and the query will be repeated. If only the repeated query appears, then there is no more information available. This facility supports only one level of additional information, so it does not help to enter more than one question mark in response to the same question.

3. Control-C (alone): This response tells the Encapsulator to back out of the question. Usually, the Encapsulator will abort the current line of questioning, back up to the previous option level, and repeat a previous question. When there is no longer any sensible place to back up to, it will end the service session and return the terminal to the NSW Front End.

4. A character string: This constitutes your answer to the question being asked. If the Encapsulator presents a list of options and if your input does not match any of the options, then the query will be repeated. You are allowed to abbreviate a reply which is a selection from an option list; however, if the abbreviation is ambiguous, it is unpredictable which option will be selected. For instance, one can say "y" or "n" to a yes/no question, but not "certainly". Of course, if the Encapsulator is not working with an option list, as when it asks for a FileSpec, then whatever you type will be accepted, whether it is valid or not.

In many cases, the encapsulator will suffix its error messages with a clause of the form " -- CODE=number". These code numbers are intended to be useful to NSW maintenance personnel in case of errors caused by system malfunctions. They are not intended to convey useful information to the NSW user.

2.2.2 Talking With The Service Program

User communication with a service program itself follows the same conventions that the particular service would use outside NSW. Usually, IBM programs respond to a subcommand language using the simple syntax defined by the IBM TSO Time-Sharing system [6]

Currently, it is not possible to interupt execution of a service program to talk with the Encapsulator, and then continue execution of the service.

2.2.3 Talking with the CCN Workspace Command Interpretor (WSCI)

The CCN Workspace Command Interpretor (WSCI) is an interactive service with the capability to execute other services sequentially. The advantages of using services through the WSCI include faster switching from service to service and the capability to pass temporary "workspace files" between services.

The WSCI uses the same conversational conventions as the CCN Encapsulator. For a more complete description of the WSCI, see the section on WORKSPACE MANAGEMENT SUPPORT.

2.3 FILES USED BY IBM SERVICES

The various files used by the various IBM services differ among one another in two primary attributes: file structure and the NSW Global File Type (GFT).

2.3.1 File Structures

There are two file structures: sequential files, which make up the large majority of NSW files, and which we call simply "files"; and library files, which are presently unique to IBM services within NSW, and which we call simply "libraries". A library is rather like a collection of (usually short) sequential files. Currently, users may create and manage libraries using the CCN Library Management (LM) support package. Under this scheme, libraries are always kept on one IBM host and cannot move about the network like ordinary NSW files. They are not used as "input" or "output" files, but rather as "direct update" files. They must always pre-exist for a service to use them, and the service always operates on the single NSW copy directly.

2.3.2 Global File Types

The NSW Global File Type (GFT) is the vehicle by which an NSW service specifies its requirements in terms of the formatting of its file data. For each file which a service requires or creates, there is an associated set of acceptable GFT's, preferentially ordered. The same GFT may be specified by any number of services.

The actual meaning of the GFT differs depending on the usage of the file:

1. OUTPUT FILES

   When a service delivers an output file to NSW, it states its GFT. The GFT describes the attributes of the data as delivered, and it remains associated with the file for as long as the file exists.

2. INPUT FILES

   As the file is being fed into the service, it is converted to one of the GFT's that the service has stated as acceptable, from whatever GFT is associated with the file itself. This conversion may be null, as when the two GFT's are the same. It may be a matter of reformatting, as when a card-image file is converted to PL/I source. It may lose information, as when a printer file is

converted to a card-image file. It may be undefined, as when one GFT represents binary data and the other represents text. Users need to know the GFT's of a service's input files in order to be able to prepare input files in a form that will not cause information to be lost when the conversion into that GFT occurs.

3. LIBRARY FILES

Under LM, libraries are not converted to the GFT requested by a service. It is your responsibility to see that the library is assigned the proper GFT when it is created. Fortunately, the CCN Library-Management services make this easy.

The following sections list GFT's that are frequently used by common CCN services and explain their primary characteristics. Notice that the notion of sequence numbers is frequently mentioned. NSW files can carry optional sequence numbers with each record, but these sequence numbers are not simply data characters within records. In order for sequence numbers to be handled properly during GFT conversion, their presence must be known to NSW. In other words, one may not just type a column of numbers in the first eight columns of a PL/I program and expect the PL/I services to treat them as sequence numbers. Unless the editor which delivers the source file to NSW declares those columns to be sequence numbers, they are going to be treated as part of the PL/I program.

Notice that all GFT names native to IBM hosts begin with the characters "360-". When speaking in the context of IBM services, we frequently elide the prefix. Thus, as used in this document, the names "360-PLI-SOURCE" and "PLI-SOURCE" are equivalent.

1. 360-KEYPUNCH

This is the standard IBM card-image file type. The records have a maximum data length of 72 characters, and longer records will be split at that length. There is an optional 8-character sequence-number field which is placed in positions 73-80. All data is forced to upper case. No format effectors of any kind are permitted. KEYPUNCH files are used for "control card decks", such as those used by the Linkage-editor services. They can also be used as source files for most language-processing services, so this GFT is used for both files and libraries.

2. 360-CARDS

This is the standard IBM card-image file type without sequence numbers. The records have a maximum data length of 80 characters, and longer records will be split. All data is forced to upper case. No format effectors of any kind are permitted. CARDS files are used for input data to PL/I programs that use "stream oriented" I/O.

3. 360-PRINT

This is the file format produced by services for transcription to a printer device with the "top-of-form" facility. All "standard system output" (SYSOUT) files are PRINT files, as are the primary informational outputs of most batch services. Since this is almost always an output type, users need not be concerned with any restrictions on its format. That is the services' problem.

4. 360-OVERPRINT

This is like 360-PRINT except that it can include printer control codes to cause overprinting. This allows, for instance, underlining.

5. 360-LIST

This is a very general format. It uses neither format effectors nor sequence numbers. It is similar to PRINT without the "top-of-form" concept, or to CARDS without the fixed-length concept. It is useful for storing machine-readable data as well as for listing on teletype-like devices.

LIST is used by most CCN services whenever text data with no particular formatting requirements is called for.

6. 360-TEXT

This is the file type for storing machine-readable data that contains lower-case alphabetics. Most IBM programs do not handle lower case, so this GFT is seldom used.

7. 360-BINARY

This is the format required by FORTRAN "unformatted I/O" statements. It can also be used by other user programs to write binary data. It is very free form, with its only major distinguising characteristic being that it represents binary data, not text.

BINARY is used by the "go" services of the native-language support packages, whenever the user indicates a binary file. In particular, any file manipulated by the FORTRAN "unformatted I/O" statements will be of type BINARY.

8. 360-OBJECT

This is the IBM binary card-image file type. It is essentially like 360-KEYPUNCH except that the 72 data columns contain eight-bit binary data. It is usually generated by a compiler service. The optional 8- character sequence-number field still contains character data. (Do not confuse this format with "column-binary" cards, which contain 80 12-bit data bytes and are not directly supported in NSW.)

9. 360-LOAD

This is another binary file type, but one of a very specific sort. It represents executable program data on the IBM system. LOAD data can only exist in a library. It is usually generated by either a compile service or a linkage-editor service. In NSW as it exists at this writing, LOAD data cannot be converted to any other type, and it cannot be transmitted between hosts.

10. 360-PLI-SOURCE

This is the file type preferred by the compiling services of the PL/I support package, so it is used for both files and libraries. More information on data of this type can be found in the section on PL/I support.

11. 360-FORTRAN-SOURCE

This is the file type preferred by the compiling services of the FORTRAN support package. More information on data of this type can be found in the section on FORTRAN support.

12. 360-ASM-SOURCE

This is the file type preferred by the compiling services of the Assembler support package. More information on data of this type can be found in the section on IBM Assembler support.

Section 3

WORKSPACE MANAGEMENT SUPPORT

3.1 USING THE CCN WORKSPACE COMMAND INTERPRETOR (WSCI)

The CCN Workspace Command Interpretor (WSCI) is a minimal implementation of the WSCI concept as documented in the NSW User Reference Manual [26]. It is merely an interactive service with the name "CCN". No actual service program is associated with the WSCI. It consists of the Encapsulator repeatedly requesting the simple name of a CCN-resident service, through the prompt string "CCN:".

For each service name entered, the WSCI invokes that service just as though it had been requested by a Front-End "USE" command. When the service is completed, instead of returning terminal control to the Front End, the Encapsulator requests the name of another local service. A service name of "QUIT," "END," or control-C terminates the WSCI and returns terminal control to the Front End.

Theoretically, every CCN interactive service is available either under the WSCI or via the USE command. In practice, there are some services which perform functions that are not useful outside the context of the WSCI, and such services are not offered via the USE command. Examples are the GET and DELIVER services. These services are all documented in this section. Services documented in other sections are available through both paths.

Notice that the WSCI executes services absolutely sequentially. It has none of the multiplexing capabilities of the NSW Front End. You can use the Front End to manage two concurrent instances of the WSCI, but each operates in its own local workspace, and there is no communication between them. In particular, workspace files created by services in one workspace are not known to services operating in another.

3.2 WORKSPACE FILES

When using the WSCI, it is possible to manipulate "workspace files" as well as NSW files. A workspace file is a temporary data collection that lasts for the life of the WSCI session only.

At this writing, the CCN encapsulator does not maintain a "Local Name Dictionary" (LND). In general, no relationship between a workspace file and an NSW file of which it is a copy is known to the encapsulator. Workspace files are either explicitly named by you, or are "unnamed", and are available only to the service that creates them.

A named workspace file must be given an IBM-compatible name, following these rules:

* The name must consists of one or more simple names, separated by periods.

* The total length of the name cannot exceed 36 characters.

* Each simple name must be from 1 to 8 characters long.

* All characters in a simple name must be alphabetic, numeric, or from the "national" set ("@", "#", "$").

* The first character of a simple name must not be numeric.

It is important to understand that a workspace is assigned to a new service session randomly. It is quite impossible to pass workspace files from one session to another without moving them into the NSW filespace.

3.3 THE CREATEW SERVICE

The CREATEW service creates a workspace file. The file survives only for the duration of the session unless you later deliver it into the NSW filespace with the DELIVER service.

You are prompted for the local file name, a GFT (default is CARDS), an approximate file size, and an optional member count. If no member count is given, the file will be sequential. Otherwise it will be a library.

Examples:

```
CCN:  createw
File name:  ?
Enter the local name of the workspace file,
or ctl-C to back out.
File name:  urge
Enter a GFT -- default is "CARDS"
GFT:  360-
File size in kilochars:  ?
Enter a rough guess of how many thousand bytes of data will
be written to this file, or ctl-C to back out.
File size in kilochars:  2
Member count, if library:  ?
Enter 0 or nothing for a sequential file; to get a library
file, enter the maximum number of members.
Member count, if library:
Workspace file urge created.

CCN:  createw
File name:  nominal
Enter a GFT -- default is "CARDS"
GFT:  360-pli-source
File size in kilochars:  5
Member count, if library:  25
Workspace file nominal created.

CCN:
```

3.4 THE DELETEW SERVICE

The DELETEW service destroys a workspace file. You are prompted for a local file name.

3.5 THE PURGEW SERVICE

The PURGEW service destroys all workspace files currently in the workspace. It requires no parameters.

3.6 THE LISTW SERVICE

The LISTW service lists the names of the files currently in the workspace. It requires no parameters.

3.7 THE EDITW SERVICE

The EDITW service is a special version of the EDIT service which operates on a workspace file as its primary input. Thus it does not involve fetching or delivering NSW files, as does EDIT. Except for this difference, its operation is the same as that of EDIT. Refer to the section on the EDIT service for more information.

Example:

```
CCN:  editw
Workspace file name:  ?
  Enter the local name of a workspace file to be edited,
  or carriage return to create a new file,
  or control-c to abort.
Workspace file name: temp2
Enter a GFT -- default is KEYPUNCH
GFT:  360-
IKJ52320I DATA SET OR MEMBER NOT FOUND, ASSUMED TO BE NEW
INPUT
00010 this is the first data record
00020 and this is the last
00030
EDIT
list
00010 THIS IS THE FIRST DATA RECORD
00020 AND THIS IS THE LAST
IKJ52500I END OF DATA
end save

CCN:
```

3.8 THE DISPLAYW SERVICE

The DISPLAYW service is a special version of the DISPLAY service which operates on a workspace file. Thus it does not involve fetching from the NSW filespace, as does DISPLAY. Except for this difference, its operation is the same as that of DISPLAY. Refer to the section on the DISPLAY service for more information.

Example:

```
CCN:  displayw
Workspace file name:  temp3
DISPLAY: 1 non
00000010   ON ENDFILE (SYSIN) NOEOD = '0'B;
00000020   DO WHILE (NOEOD);
00000030      RILE (SYSIN) INTO (BUFFER);
00000040      IF NOEOD
00000050         THEN  GET FILE (SYSPRINT)
00000060                  EDIT ('DATA READ:  ' || BUFFER)
00000070                       (SKIP, A);
00000080         ELSE  GET FILE (SYSPRINT)
00000090                  EDIT ('END OF DATA')
00000100                       (SKIP, A);
00000110   END;
DISPLAY: end

CCN:
```

3.9 THE GET SERVICE

The CCN GET service makes a copy of an NSW file in the service session workspace. This copy survives only for the duration of the service session.

You are prompted for an NSW filespec, a simple name for the workspace copy, and an optional NSW Global File Type (GFT). If you specify a GFT, it MUST be one of the CCN-native types; therefore, you do NOT type the "360-" prefix. If you do not specify a GFT, then GET assigns one. If the GFT of the NSW file is native to CCN (begins with "360-"), then it is used. Otherwise, either "TEXT" or "BINARY" will be used.

Unless you have a good reason for specifying a GFT, it is wise to let GET select one.

Example:

```
CCN:  get
NSW filespec: ...at-ccn
Enter a GFT -- default is file's current type
GFT:  360-
Set a lock? (yes or no): n
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.AT-CCN
[now talking to 2ccn (...ccn)]
Workspace file name: temp2

CCN:
```

3.10 THE DELIVER SERVICE

The CCN DELIVER service moves a file from the service session workspace into an NSW file, deleting it from the workspace.

You are prompted for the simple name of the workspace copy, the NSW file name, and an optional NSW Global File Type (GFT) to be assigned. If you specify a GFT, it MUST be one of the CCN-native types; therefore, you do NOT type the "360-" prefix. If you do not specify a GFT, then DELIVER assumes "TEXT".

If the file includes sequence numbers, you should specify a GFT; otherwise, the sequence numbers will be treated as part of the data.

Example:

```
CCN:  deliver
Workspace file name:  ?
Enter the simple name of the workspace file to be delivered
   or control-c to abort.
Workspace file name:  hydra
Enter a GFT -- default is "LIST"
GFT:  360-
NSW filespec:  ?
Enter the NSW file name to be assigned the delivered copy
   or control-c to abort.
NSW filespec:  hydra.listing
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.HYDRA.LISTING
[now talking to 2ccn (...ccn)]
New filename is $CCN.HYDRA.LISTING

CCN:
```

3.11 THE TSO SERVICE

The CCN TSO service is provided primarily as a convenience to NSW implementors. It is not intended to be a full-facility access path to TSO, and NSW users use it only at their own risk.

The TSO service switches the WSCI from the mode where WSCI commands are requested by the "CCN:" prompt to a mode where IBM TSO Commands are requested by the "TSO:" prompt.

In this mode, the user operates entirely without NSW services, and in a temporary workspace. Therefore, NO FILES are available except those workspace files that were set up through WSCI commands entered before entering the TSO service. Likewise, NO FILES can be saved except through WSCI commands entered after the TSO service is exited.

Like the WSCI, the TSO service is exited by entering "END", "QUIT", or control-C.

Except for the constraints of the temporary workspace, the TSO service is very like using IBM TSO outside NSW, with the "TSO:" prompt taking the place of TSO's usual "READY" prompt. However, only TSO command-processor programs can be invoked. This excludes any use of command lists, even system-provided ones. It also excludes the use of those commands that are built into the TSO Terminal Monitor Program, such as TEST, CALL, TIME, and EXEC.

Section 4

NSW FILE MANAGEMENT SUPPORT

4.1 THE DISPLAY SERVICE

The DISPLAY service is an NSW offering of the CCN DISPLAY processor as implemented under the CCN TSO system, but operating under the CCN NSW Encapsulating Foreman. DISPLAY is used to browse text files, particularly the outputs of batch jobs. It is not a text editor, and it does not have the capability to alter data; for that purpose, use the EDIT service.

The primary purpose of DISPLAY is to view the outputs of CCN-based NSW batch services, which are normally stored on the CCN system. DISPLAY is not sensitive to the NSW Global File Type (GFT) of its input files, except to the text/binary attribute, so it can be used to browse any non-binary NSW file; however, initial response time will suffer if the file has to be fetched from another NSW host.

4.1.1 Invoking DISPLAY

When you type the USE command that invokes the DISPLAY service, these events should occur:

1. Your terminal is connected to the CCN Encapsulator.

2. The Encapsulator queries you for the identity of a file to be browsed. If you enter control-C, that is interpreted as a request to terminate the session, and you will skip to step 6.

3. A copy of the file is made available to the service. If there is a local NSW copy that can be browsed directly, it will be used, since DISPLAY does not have the capability to destroy data. In any other case, a service copy must be created in the service's workspace.

4. Your terminal is connected to the DISPLAY program. The two of you cooperate in browsing the file.

5. Eventually you issue the END subcommand. DISPLAY terminates, and your terminal is reconnected to the Encapsulator. The session reverts to step 2.

6. When you have entered control-C, your terminal is once again connected to the NSW Front End, where you issued the USE command.

7. The Front End prompts you for another NSW command.

Your communication with the DISPLAY program itself is limited to entering the simple subcommands described in the next section. On occasions, you will also wish to abort a subcommand, as for instance a LIST subcommand, by using the "interrupt-process" character (control-C).

Whenever DISPLAY is ready to accept a new subcommand, it prompts with the string "DISPLAY". You can then enter one of the commands DEFAULT, END, FIND, LIST, or PAGE.

4.1.2 Subcommands

DISPLAY subcommands follow the syntax of the TSO command language as defined in [6]. A DISPLAY subcommand consists of a subcommand name, followed by an optional list of operand strings separated by blanks and/or commas, and terminated by a semicolon or carriage return. An operand string is usually either a simple number or quoted string or a keyword name optionally followed (without a blank) by a parenthesized list. The list follows the same syntax as the main operand list except that it is terminated by its closing parenthesis instead of a semicolon or carriage return. Subcommand names and keyword operands may be abbreviated to any unambiguous leading string. Blanks and/or commas are used to separate operands, and should not occur in any other place except within quoted strings.

In the descriptions below, the full syntax of each DISPLAY subcommand is shown in the form of a skeleton subcommand. In these skeletons, operands that are given as lower-case strings in single quotes represent "positional" operands -- operands whose order is critical (but some of them may be optional, nevertheless). You are to substitute some value for the entire quoted string (the quotes are part of the metalanguage). Operands shown as upper-case strings, sometimes with parenthesized lists, represent "keywords" -- operands that may be entered in any order, provided they follow all positional operands.

Keyword operands separated by "/" represent different names, with different meanings, for the same basic option. Only one should be used in a given subcommand.

4.1.2.1 The DEFAULT Subcommand

The DEFAULT subcommand is used to set or reset default display control options. Its effects remain until the next DEFAULT subcommand is issued. Various other DISPLAY subcommands carry some of the same operands as DEFAULT, and with the same meanings; however, such options only hold for the duration of the subcommand on which they are stated. When such a subcommand does not have one of these options stated, its value is taken from the last DEFAULT subcommand that was issued.

The full syntax of DEFAULT is:

```
DEFAULT COMPRESS/NOCOMPRESS
   NUMBER/NONUMBER
   WIDTH('number')
   COLUMN('number')
```

No operands are required. Those not explicitly specified remain as before the DEFAULT subcommand was issued. Before any DEFAULT is issued, the control options are set to:

```
DEFAULT NOCOMPRESS NUMBER WIDTH(999) COLUMN(1)
```

The meanings of the options are:

COMPRESS or NOCOMPRESS

> If COMPRESS is specified, multiple blanks will be compressed to a single blank. The compression is done AFTER the COLUMN and WIDTH options have been applied to the output line.

NUMBER or NONUMBER

> If NUMBER is specified, a line number is printed to the left of each output line. Unless the input file is organized into printer pages, this is a single logical record number. If the file is organized into printer pages, it is of the form 'page-number':'line-number'. Overprinted lines will be considered to be separate lines.

WIDTH ('number')

> The maximum number of characters printed per record displayed is set to 'number'. COLUMN and WIDTH are independent.

COLUMN ('number')

> When a record is displayed, it will be printed starting at column 'number'. COLUMN and WIDTH are independent.

4.1.2.2 The END Subcommand

The END subcommand is used to stop browsing the current file and return control to the Encapsulator. It has no operands -- just type "END".

4.1.2.3 The FIND Subcommand

The FIND subcommand is used to display the records in which a specified string occurs. You may find a single such record, the next such record after the last one found, or all such records. The search may be over the entire file or restricted to a certain part of the file, and it may be confined to certain columns of the logical records.

The full syntax of FIND is:

```
FIND 'starting record number'
   'ending record number'
   'search string'
   ASIS
   BETWEEN('col1','col2')
   COUNT('number')/K('number')/ALL
   COMPRESS/NOCOMPRESS
   NUMBER/NONUMBER
   WIDTH('number')
   COLUMN('number')
```

The 'search string' operand is required except in one special case: if a FIND operation terminates because the COUNT is exhausted, it may be resumed with an assumed COUNT of one by entering FIND with no operands. The defaults for omitted operands are:

```
1 999999 COUNT(1) BETWEEN(1,999)
```

The meanings of the options are:

'starting record number' and 'ending record number'

> These numbers specify the range within which the search is to be done. Unless the file is organized into printer pages, each is entered as one- to six-digit integers. If the file is organized into printer pages, each is entered as a pair of one- to three-digit integers in the form 'page-number':'line-number', or as a single page number. If a page or line number is greater than that of any page or line in the file or in the

indicated page, then the last record of the file or page is assumed.

'search string'

This string is enclosed in arbitrary matching delimiters (see section "Typical Commands"). It will match only an "identical" string which occurs completely within the bounds of a record. Unless the ASIS operand is specified, differences in alphabetic case will not prevent matches between the search string and file data.

ASIS

This operand causes differences in alphabetic case to prevent matches between the search string and file data.

BETWEEN('col1','col2')

This operand specifies an inclusive range of columns within which the search is to be confined.

COUNT('number'), K('number'), or ALL

This operand specifies the maximum number of records to be displayed before the FIND operation is interupted. "K" is an alias for "COUNT". "ALL" is equivalent to "COUNT(999999)".

The remainder of the options are the same as those for the DEFAULT subcommand. If specified here, they will have effect only for the duration of the FIND operation.

4.1.2.4 The LIST Subcommand

The LIST subcommand is used to display all or a part of the File. The full syntax of LIST is:

```
LIST 'starting record number'
   'ending record number'
   COUNT('number')/K('number')
   COMPRESS/NOCOMPRESS
   NUMBER/NONUMBER
   WIDTH('number')
   COLUMN('number')
```

The defaults for omitted operands are:

1 999999 COUNT(999999)

The meanings of the options are:

'starting record number' and 'ending record number'

These numbers specify the range of records which are to be displayed. Unless the file is organized into printer pages, each is entered as one- to six-digit integers. If the file is organized into printer pages, each is entered as a pair of one- to three-digit integers in the form 'page-number':'line-number', or as a single page number. If a page or line number is greater than that of any page or line in the file or in the indicated page, then the last record of the file or page is assumed.

COUNT('number') or K('number')

This operand specifies the maximum number of records to be displayed as a result of this LIST operation. "K" is an alias for "COUNT".

The remainder of the options are the same as those for the DEFAULT subcommand. If specified here, they will have effect only for the duration of the LIST operation.

4.1.2.5 The PAGE Subcommand

The PAGE subcommand is used to display a particular page of the file. If the file is not organized into printer pages, then PAGE will display a particular line. The full syntax of PAGE is:

```
PAGE 'page number'
   COUNT('number')/K('number')
   COMPRESS/NOCOMPRESS
   NUMBER/NONUMBER
   WIDTH('number')
   COLUMN('number')
```

The defaults for omitted operands are:

1 COUNT(999999)

The meanings of the options are:

'page number'

This number specifies the page to be displayed. If the file is not organized into printer pages, then it is interpreted as a line number.

COUNT('number') or K('number')

This operand specifies the maximum number of records to be displayed as a result of this PAGE operation. "K" is an alias for "COUNT".

The remainder of the options are the same as those for the DEFAULT subcommand. If specified here, they will have effect only for the duration of the PAGE operation.

4.1.3 Typical Commands

The following are sample DISPLAY subcommands:

```
DEF C(9) W(62)
                All subsequent DISPLAY operations will
                display only columns 9-70.
D COMP NON
                All subsequent DISPLAY operations will
                show compressed text without line
                numbers.
F 'bursitis' ASIS ALL
                Display all occurrences of the string
                "bursitis" in the entire file.  Do not
                consider upper-case letters to match
                (e. g., "Bursitis" is not a match).
FIND 5:9 /twister/ K(7) BETW(10 20) COL(1) W(8)
                Locate the first seven logical records,
                at or after the ninth one on page 5,
                which contain the string "twister", in
                any combination of upper and lower case,
                within columns 10 to 20.  For each such
                record found, display columns 1 through 8.
L
                List the entire file.
LIST 9:34 10:7 COMP
                List the contents of the file from page
                9, line 34 through page 10, line 7.
                Compress out excess blanks.
P 7
                List the contents of page 7.
END
                Return to the NSW Encapsulator.
```

4.1.4 Sample Service Session

```
CCN:  display

Filespec:  ?
Enter the NSW file specification of a file to be
  browsed, or a control-c to terminate the tool.

Filespec:  ...body
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.BODY
[now talking to 2ccn (...ccn)]
DISPLAY: help

SUBCOMMANDS
     LIST,PAGE,FIND,DEFAULT,END
IKJ56804I FOR MORE INFORMATION
          ENTER HELP SUBCOMMANDNAME OR HELP HELP
DISPLAY: list non
00000010   ON ENDFILE (SYSIN) NOEOD = '0'B;
00000020   DO WHILE (NOEOD);
00000030      READ FILE (SYSIN) INTO (BUFFER);
00000040      IF NOEOD
00000050         THEN PUT FILE (SYSPRINT)
00000060                  EDIT ('DATA READ:  ' || BUFFER)
00000070                       (SKIP, A);
00000080         ELSE PUT FILE (SYSPRINT)
00000090                  EDIT ('END OF DATA')
00000100                       (SKIP, A);
00000110   END;
DISPLAY: f 'put' all comp
000005 00000050 THEN PUT FILE (SYSPRINT)
000008 00000080 ELSE PUT FILE (SYSPRINT)
DISPLAY: end

Filespec:  ?
Enter the NSW file specification of a file to be
  browsed, or a control-c to terminate the tool.

Filespec:  @
!
ending DISPLAY session
```

4.2 THE EDIT SERVICE

The EDIT service is an NSW offering of the IBM TSO EDIT processor [6] as implemented on the CCN computing system. As an NSW service, it operates under the CCN NSW Encapsulating Foreman. This section is a brief description of EDIT that uses NSW parlance and excludes those features of the editor that are not available from the encapsulated version.

The EDIT service is used to create, modify and examine NSW files. Through this service, data is entered into the system from the NSW terminal and stored into new or existing NSW files. Such data may comprise source programs, text, data used as service input, or any other non-binary data. EDIT is not particular about the NSW Global File Type (GFT) of its input file, but it will deliver its output under the same type unless you choose to specify a different type. If you are creating a new file, you will have to specify a type.

While EDIT can also be used to browse a file without altering it, the DISPLAY service is more suitable for this purpose.

EDIT can be used as an NSW service, or as a command of the CCN WSCI. The differences between the two usages are illustrated in Figure 1.

1. When EDIT is invoked directly through the "USE ...EDIT" command, only one NSW file can be edited in a single service session, and not more than one file can be delivered back to NSW as a result of a single session. The input file can be an existing file or a new file. Delivery can replace an existing file or create a new NSW file. If the EDIT session does not produce a useful edited file, delivery can be skipped.

2. When EDIT is invoked under the CCN WSCI, that is, through "USE ...CCN", more capability is available. You can invoke the WSCI GET command to place named copies of your NSW files into your session workspace. Then you can invoke the WSCI EDIT command to operate on a primary NSW file and on those workspace files, to save the results in one or more workspace files, and optionally, to deliver the primary workspace file back to NSW. Then you can use the WSCI DELIVER command to move any secondary workspace files back into the NSW file space. Notice that there is no capability for interrupting EDIT, reverting to the WSCI, and then resuming EDIT. It is possible to end EDIT, perform WSCI work, and then start EDIT again from scratch; however, restarting EDIT requires specifyingy a new NSW file name, not a workspace file name.

In either case, you should be aware of your exposure to loss of data should any systems crash occur. While such crashes are infrequent, massive editing is still safest done in several short sessions rather than one very long one. It is not sufficient to use occasional SAVE subcommands to checkpoint the editing process, since only *delivered* NSW file data will survive a system crash.

In keeping with its IBM heritage, EDIT is oriented toward sequenced files. It is possible to edit unsequenced data, but it is not convenient. There is no direct connection between NSW file sequence numbers and those used by EDIT. EDIT merely expects to find sequence numbers, if any, in either the first eight or the last eight characters of each data record. The NSW file descriptions of IBM data types have been designed to make this happen in virtually all cases, and you shouldn't have to worry about it. Still, it is hoped that a future version of EDIT will have a more definite understanding of NSW file sequence numbers.

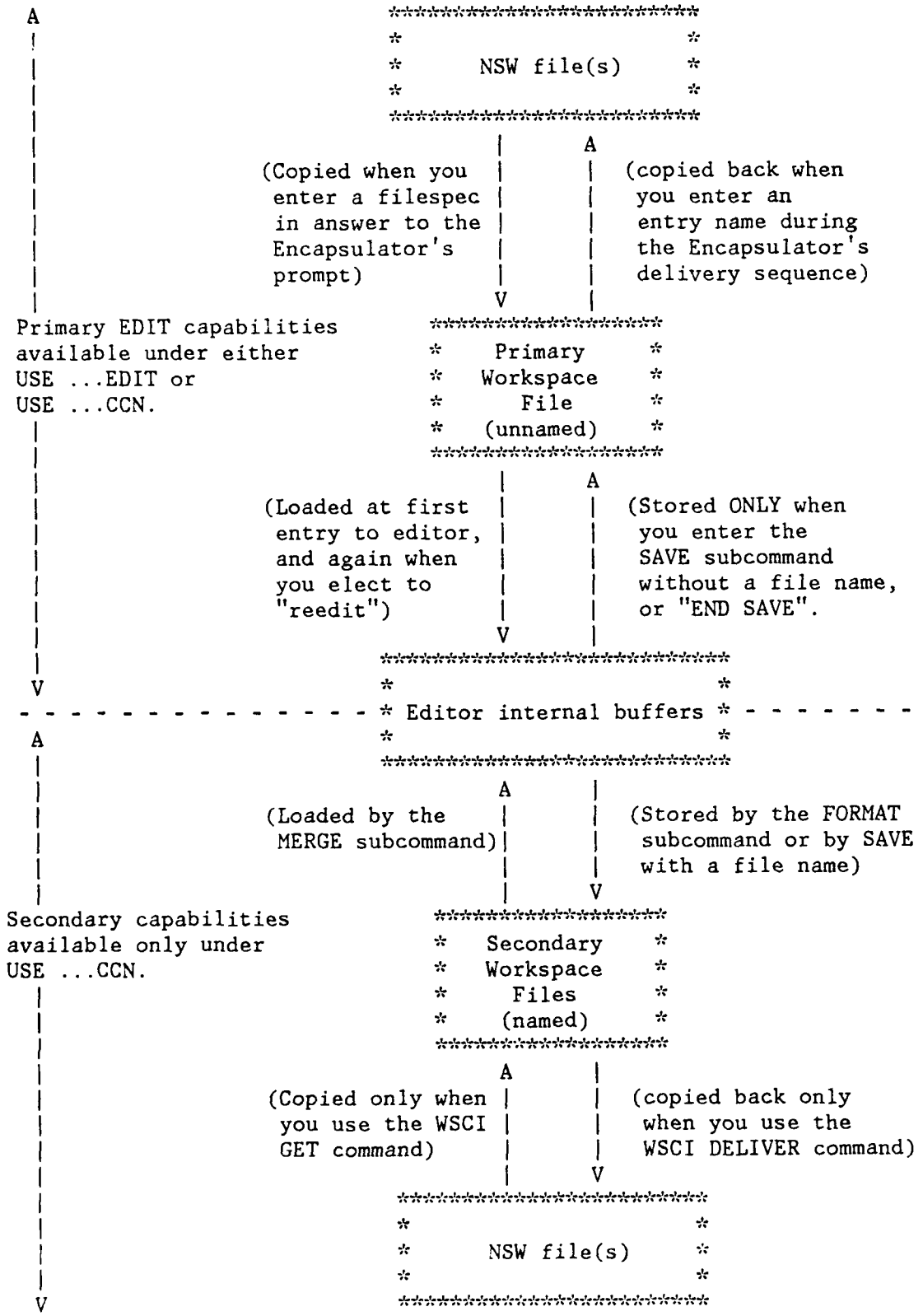


Figure 1: How EDIT Uses Files

4.2.1 Invoking the EDIT Service

When you type "USE ...EDIT", the following events should occur:

1. Your terminal is connected to the CCN Encapsulator.

2. The Encapsulator queries you for information about the file to be edited. It establishes whether it is an old file or a new one, and what basic type of file it is.

3. A copy of the file (empty if new) is made in the service's private workspace.

4. Your terminal is connected to the IBM TSO EDIT program, and that program loads its internal buffers from the workspace file. The two of you do the kinds of things that users and editors do to data. Usually, unless you decide to abandon the edit session, this includes one or more SAVE subcommands. Whenever you issue SAVE, and at no other times, the editor's internal buffers are rewritten to the workspace file.

5. Eventually you issue the END subcommand. The editor terminates, and your terminal is once again connected to the Encapsulator.

6. The Encapsulator gives you the opportunity to "re-edit", that is, to loop back to step 4, and edit the current contents of the workspace file.

7. When you do not elect to re-edit, the Encapsulator queries you about how to deliver the edited data. You can skip delivery, deliver to a new NSW file, or, if you were editing an existing NSW file, deliver back to the input file.

8. The Encapsulator tells you the full name of the NSW file to which delivery was made.

9. Then it reconnects your terminal to the NSW Front End, where you issued the USE command. The Front End reports to you the resources used by the entire service session, and prompts you for another NSW command.

The relationship between an NSW file, the NSW service workspace copy, and the editor's internal buffers is illustrated by figure 1.

4.2.2 Invoking the WSCI EDIT Command

When you type "EDIT" as a CCN WSCI command, the same basic events occur. However, since you are already in conversation with the CCN Encapsulator, step 1 has already happened, and step 9 is deferred until you end the WSCI session.

4.2.3 Subcommand Modes

The editor is always operating in one of two subcommand modes: "edit" or "input". In edit mode, you can only enter editor subcommands. In input mode, everything that you type is entered into the file as data. In either mode, you can always switch to the alternate mode just by typing a bare carriage return. Usually, though, you will want to enter input mode through the INPUT subcommand, since it allows you to specify a new point within the file where your new data is to be placed. You can also enter edit mode through the "interrupt-process" character, control-C.

The two modes are readily distinguished by the prompt strings which the editor displays when it changes mode. When entering edit mode, the string "EDIT" is displayed. This string is also used as a prompt for the next subcommand in certain situations in edit mode; however, no prompt is issued after subcommands that cause data display, and certain trivial subcommands (as VERIFY) cause neither display nor prompt.

In the input mode, if the file is sequenced, data line entry is prompted by the sequence number of the line to be entered. If the file is not sequenced, there is not prompt in input mode. However, in either case, the string "INPUT" is displayed when the editor switches from edit to input mode.

Usually, when you enter the editor, you will begin in edit mode; however, if the file is empty (as when it is a new file), you will begin in input mode.

4.2.4 Continuation Lines

Whenever a line entered to the editor ends with a hyphen immediately followed by a carriage return, a "continuation line" is indicated. The editor assumes that the carriage return is for the convenience of the terminal user, and does not signify end-of-line. Both the hyphen and the carriage return are discarded by the editor, and subsequently typed characters become a part of the subcommand or data line that was being typed when the hyphen was entered. If you need to create a data line that ends in a hyphen, follow it by at least one blank. If that method is not applicable to your situation, you can probably use the CHANGE subcommand.

4.2.5 Specifying Lines and Ranges

The editor maintains a virtual "current line pointer" which always indicates the text line that was last operated on. In some of the editor's subcommands, unless a specific line number is stated, the "current line" is implied. The current line can usually be referenced explicitly by using a "line number" of "*". See [6] for a chart of the effects of various editor subcommands on the pointer.

The scope of an editor subcommand is frequently stated in terms of a "line specification" which may indicate either a single line or a group of contiguous lines. There are two basic forms of line specifications:

1. An explicit specification is one or two line numbers, specifiying either the only line or the first and last lines of a contiguous range. This form can only be used for numbered files.

2. A context specification consists of the current line indicator "*", optionally followed by a number indicating how many contiguous lines beginning at the current line are being specified. This form can be used for numbered or unnumbered files.

4.2.6 Verify Modes

The editor is always operating in one of two "verify" modes, called "verify on" and "verify off". These modes govern how much feedback is given the user as his subcommands are executed. When verify is off, this feedback is minimal. When verify is on, EDIT displays the current line whenever:

1. The position of the current line pointer changes, as, for example, the result of an UP, DOWN, or FIND subcommand. (A minor exception arises with the TOP subcommand, which sets the current line pointer to "0". Unless a line 0 actually exists in the file, no current line display occurs.)

2. The text or sequence number of the current line is changed (for example, by a CHANGE or RENUM subcommand).

The default verify mode is ON. Turning it OFF is not reccommended if the file is unnumbered.

4.2.7 Subcommands

4.2.7.1 Summary

A complete description of the subcommands of EDIT can be found in [6]; however, it is quite possible to use the editor effectively given only its online HELP facility. This facility will print for the online user a list of available subcommands or a description of the function, syntax, and operands of a given subcommand. It will also reflect any embellishments that CCN may have made, which are not reflected in [6]. For a descripton of the usage of HELP itself, enter the subcommand "HELP HELP".

In the table below, note that the aliases used by EDIT for UP, DOWN, and TOP are not those specified in [6]. CCN has changed these aliases for compatibility with previous systems. In any case of conflict between this document and [6], this document should take precedence.

The following subcommands are available through EDIT:

Subcmd:	Alias:	Function:
ALLOCATE	ALLOC	creates an empty workspace file.
BOTTOM	B	Points to the last line of the file.
CHANGE	C	Modifies character strings.
COPY	CO	Copies a line or a range of lines from one portion of the file to another.
DELETE	DEL	Deletes lines from the file.
DOWN	D	Points closer to the end of the file by a specified number of lines.
END		Returns to the NSW service Encapsulator.
FIND	F	Locates a specific character string in the file.
FORMAT	FORM	Formats all or part of the file, either to the terminal or to a workspace file. If the subcommand is used to create a workspace file, its name will be suffixed by ".LIST" and its type will be "360-PRINT".
HELP	H	Displays information on subcommands.
INPUT	I	Enters input mode.
INSERT	IN	Inserts new lines of data.
"linedit"		Inserts, replaces, or deletes a single line merely by typing its line number (or "*") followed (except for delete) by a blank and the line's data. This function does not actually have a subcommand name, but it is known to the HELP facility as "linedit".
LIST	L	Displays one or more lines of the file at the terminal.
MERGE	M	Merges data from a workspace file into the editor's buffers. The file will usually have been created by the GET service. The file being edited can be specified by "*".
MODIFY	MOD	Alters the contents of a line by replacing, deleting, or inserting characters.
MOVE	MO	Moves a line or range of lines from one portion of the file to another. The original line(s) are deleted.
PROFILE	PROF	Specifies non-standard character- and line-deletion characters.
RENUM	REN	Sequence-numbers the file.
SAVE	S	Stores the edited file back into the service workspace copy, or into another workspace file.
TABSET	TAB	Establishes or changes tab stops for use in data entry.
TOP	T	Points to line 0, whether it exists or not.
UNNUM		Removes sequence numbers from a numbered file.
UP	U	Points closer to the beginning of the file by a specified number of lines.
VERIFY	V	Turns verify mode on or off.

4.2.7.2 Unavailable Subcommands

When using HELP or consulting [6], you will see references to some subcommands which are unavailable or not useful from EDIT, usually because of the constraints of the NSW environment. Some of these subcommands may be supported in a future version of EDIT; however, in the present version, their use may cause unpredictable and nonrepeatable results, possibly including abnormal termination of the service.

These subcommands are:

DO
EXEC
FSE
RUN
SCAN
SEND
SUBMIT

4.2.8 Example

```
CCN:  edit
Filespec: ?
Enter the NSW file specification of a file to be edited,
  or carriage return to create a new file,
  or control-c to abort.
Filespec: ...testl.body
Enter a GFT, or carriage return to use the file's current type
GFT:  360-
Set a lock?  ?
Do you intend to replace the same NSW file with the edited data (Yes or
Set a lock?  n
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.BODY
[now talking to 2ccn (...ccn)]
EDIT
help
SUBCOMMANDS -
  ALLOCATE,BOTTOM,CHANGE,COPY,DELETE,DO,DOWN,END,EXEC,FIND,FORMAT,FSE,
  HELP,INPUT,INSERT,LIST,MERGE,MODIFY,MOVE,PROFILE,RENUM,RUN,SAVE,
  SCAN,SEND,SUBMIT,TABSET,TOP,UNNUM,UP,VERIFY. (FOR EXPLANATION
  OF LINE INSERT/REPLACE/DELETE FUNCTION, ENTER HELP LINEDIT).
IKJ56804I FOR MORE INFORMATION ENTER HELP SUBCOMMANDNAME
          OR HELP HELP
list
00010    ON ENDFILE (SYSIN) NOEOD = '0'B;
00020    DO WHILE (NOEOD);
00030       READ FILE (SYSIN) INTO (BUFFER);
00040       IF NOEOD
00050          THEN PUT FILE (SYSPRINT)
00060                   EDIT ('DATA READ:  ' || BUFFER)
00070                        (SKIP, A);
00080          ELSE PUT FILE (SYSPRINT)
00090                   EDIT ('END OF DATA')
00100                        (SKIP, A);
00110    END;
IKJ52500I END OF DATA
top
c * 999 /PUT/ GET/ all
00050          THEN  GET FILE (SYSPRINT)
00080          ELSE  GET FILE (SYSPRINT)
mod 30
00030       READ FILE (SYSIN) INTO (BUFFER);
MODS->       <<<<<
00030       RILE (SYSIN) INTO (BUFFER);
end save
Re-edit?  ?
Do you wish to edit the same data again
 from the last SAVE subcommand?
  (YES or NO -- default is 'NO')
Re-edit?  n
delivery type = NEW/QUIT/SAME/WORKSPACE:  new
```

```
NSW filename:  edited
Pick a delivery GFT (default is PLI-SOURCE)
GFT:  360-
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.EDITED
[now talking to 2ccn (...ccn)]
new NSW file is $CCN.EDITED

CCN:
```

4.3 THE SEND SERVICE

The CCN SEND service is provided primarily as a convenience to NSW implementors. It is not fully supported by internal NSW protocols, and NSW users use it only at their own risk.

The SEND service is like the NSW COPY command. It copies the data from one NSW file into another. However, the SEND command allows you to specify the NSW host on which the copy is to reside, and the Global File Type (GFT) to be assigned it.

When you use SEND, you are prompted for the input filespec, the output file name, the receiving host name, and the optional GFT.

You can specify no GFT, in which case the file's existing GFT will be used. This will only work if the sending and receiving host are of the same NSW host family.

If you specify a GFT, it must be native to the receiving host's NSW family. Even if that host is CCN, you cannot ellide the host-family prefix. For instance, if the receiving host is CCN, you can say "360-TEXT", but "TEXT" will not work. The SEND command makes no attempt to validate your GFT before attempting to make the copy.

Example:

```
CCN:  send
Old filespec:  ?
Enter the NSW file specification of the file to be copied
   or control-c to abort.
Old filespec:  at-ccn
New filespec:  ?
Enter the NSW file specification of the file to be created
   or control-c to abort.
New filespec:  sent.file
Destination Host:  ?
Enter the NSW host name where the copy is to be stored
   or control-c to abort.
Destination Host:  radc-20
New GFT: " , -
Enter a GFT to assign the copy,", -
   or null to use the existing GFT,", -
   or control-c to abort.", -
   Note: no local validation can be done on any GFT you enter.
New GFT:  10x-text
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.AT-CCN
[now talking to 2ccn (...ccn)]
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.SENT.FILE
[now talking to 2ccn (...ccn)]
File at-ccn copied to sent.file at radc-20
```

Section 5

LIBRARY-MANAGEMENT SUPPORT

5.1 WHAT IS A LIBRARY?

A library is a collection of related sequential files, called "library members", which is allocated to a requesting service as a unit. The service then selects and reads members as if they were separate files. The name of the library does not matter to the service, since you supply it, in one way or another, when you start the service; however, the names of the members are known to the service, and members are sought by the service by name. The library thus represents a "search scope" within the NSW filespace. It also represents a private sub-namespace within which both a service and a user refer to members by the same simple names.

Library Management, or LM, is a support-function package designed to make the most essential features of IBM library file support available to the NSW user.

5.2 WHEN WILL YOU USE A LIBRARY?

When you use services mounted on an IBM System/360 or System/370 system, you will have many occasions to use libraries. Theoretically, a library can have any Global File Type (GFT); however, in practice, only four types of libraries are usually needed: "card-image" libraries, "PL/I source" libraries, "load-module" libraries, and "object" libraries. You will use them in a least the following cases:

5.2.1 Load-Module Libraries

A load-module library is of GFT 360-LOAD.

Any executable program on an IBM system is invoked by a simple name, under an environment that states or implies a search scope. Therefore, executable programs, or "load modules", are members of load module libraries. In fact, the format of a load module precludes storing it in any other manner, so that you cannot put a single load module into a sequential NSW file.

Obviously, any NSW IBM service that produces a binary IBM program as its output will store it in a load-module library. This includes most of the services in the CCN Native Language Processing (NLP) support packages. If you are going to save any binary programs, you will need at least one load-module library. You create and maintain the library with the LM support package, but you create and use members with NLP services.

5.2.2 PL/I Source Libraries

A PL/I source library is of GFT 360-PLI-SOURCE.

The PL/I %INCLUDE statement fetches a segment of PL/I source code by name, from some user-specific search scope. Therefore, PL/I "canned" text segments are stored in libraries. If you are going to use %INCLUDE, you will need at least one PL/I source library. You create and maintain both the library and its members using the LM support package and an editor service. Only services in the CCN PL/I support package will actually "use" the library.

5.2.3 Card-Image Libraries

A card-image source library may be of GFT 360-CARDS, 360-KEYPUNCH, or 360-ASM-SOURCE, depending on how specific you want to imply the contained data to be formatted.

The COPY statement and all the macro-assembly features of the IBM assembler fetch code segments by name, from some user-specific search scope. Since the IBM assembler demands "card-image" input, all such text is stored in card-image libraries. If you use the COPY statement, or if you define your own macro definitions (except for those local to and contained within a single source program), you will need at least one card-image library. You create and maintain both the library and its members using the LM support package and an editor service. Only services in the CCN Assembler support package will actually "use" the library.

5.2.4 Object Libraries

An object library is of GFT 360-OBJECT.

Most CCN language-support packages allow you to deal only with source-language files or load-module files; however, there is an intermediate card-image binary form in which code can be stored. This form, called "object" form by IBM, must be used by certain cross-compiler systems which do not ever produce code to be loaded into an IBM computer, and

which therefore have no reason to produce load modules. To support these compilers, the LM package supports Object libraries.

5.3 CHARACTERISTICS OF IBM LIBRARIES

Under LM, the library owner must perform certain housekeeping chores to keep a library tidy, so he must necessarily understand some of the characteristics of the IBM library implementation.

5.3.1 Host Dependence

Under LM, only one physical copy of a library can exist, so the library data cannot be made directly available to a service on another host. The IBM support package provides services to copy members from certain library types into NSW sequential files, and vice-versa, and such sequential files can be used in any way that any NSW file can; however, they remain separate files, unrelated to the library except in the user's mind.

5.3.2 Member Names

Each member has a simple name, and may have a number of "aliases". An alias is just like a true member name in almost every respect. An exception is that the load-module attributes stored in the directory of a load-module library may differ among aliases and true names. This facilitates describing subroutines with multiple entry points. Under LM, member and alias names follow a standard IBM syntax, that is:

1. A member name is 1 to 8 characters long.
2. All characters are alphabetic, numeric, or national (the "national" characters are "@", "#", and "$").
3. The first character is not numeric.

5.3.3 Library Representation

A library is usually created by the CREATEL service. At that time, it is allocated a fairly fixed piece of disk storage. The total space available for the library can wax and wane, within limits, according to library use; however, the extent to which this can actually happen is governed by conditions external to NSW and not under the library owner's

control. Therefore, it is important, when creating a library, to make a reasonable estimate of its "average" size. This need not accomodate the maximum expected size, but it should not differ from it by an order of magnitude. See the section on "GARBAGE COLLECTION" for more information on size estimation.

At the time of library creation, an unnamed member called the "directory" is created. This serves as the "table of contents" for the library, and in the case of a load-module library, it holds the set of member attributes that a loader program would use to allocate main-storage space for a program (which is why a load module cannot exist outside a library). Once created, the directory is absolutely fixed in size, so you must make a good estimate of the *maximum* number of members and aliases that the library will ever need to hold. Directory space is fairly cheap, so you can afford to overestimate somewhat here. If the directory ever overflows, you can reclaim space in it by deleting unneeded members. If all members are valuable, you will have to create a new library, with a larger directory, and merge the old one into it.

5.3.4 Garbage Collection

The space in a library is allocated sequentially. When a member is "overwritten", the new data is actually added to the end of the file, and the original becomes wasted space. Likewise, when a member is deleted, its space is not immediately reclaimed. If this kind of activity is frequent, a library can outgrow its space allocation very quickly. For this reason, you might be well advised to plan for an "average" file size about twice what you expect to actually store in the library.

The LM support package provides a "COMPRESS" service which performs garbage collection on a library, and a "LIBMAINT" service which can help determine when this should be done. You should certainly compress a library whenever an operation that would have added data to it fails. Normally, you will want to do it more often, to prevent the failure of time-consuming operations.

5.4 SERVICE SUMMARY

The LM support package provides basic services for creating, merging, and compressing libraries. There is one truly interactive service, "LIBMAINT", which accepts a subcommand language which is detailed below. The other services are not interactive except during the specification of their parameters; however, all the services execute in the foreground, under the interactive Encapsulator.

5.5 THE CREATEL SERVICE

CREATEL creates a library in the NSW file space. You are prompted for the NSW file name, the GFT, the approximate file size in thousands of characers, and the approximate maximum number of members that the directory needs to be able to accomodate. On successful competion of the service, the library has been entered into the NSW file catalog, and is ready to be used by other services.

Remember to inflate your size estimate to allow for the space lost in normal file update. The particular allocation scheme used will allocate one half the stated number of characters (very approximately) of disk space, with provision for (under optimal conditions) expansion to approximately eight times the stated size.

5.6 THE PUTMEM SERVICE

PUTMEM copies the data from a sequential file into a member of an existing library. If the member already exists, it is replaced. Otherwise, it is created. PUTMEM can operate on any library except a load-module library.

5.7 THE GETMEM SERVICE

GETMEM copies the data from a member of an existing library into a sequential NSW file. It can operate on any library except a load-module library.

5.8 THE LIBMAINT SERVICE

The LIBMAINT service allows you to access and manipulate the directory and selected members of a library. For load-module libraries, where certain status information is stored in the directory itself, some extended functions are provided. LIBMAINT is an interactive service that functions as a subcommand processor. It prompts for each subcommand with the herald string "PDS:" ("PDS" is IBM parlance for "library", but you might think of it as standing for "Please Demand Something"). The available subcommands, which are described in detail below, are:

```
ALIAS    DISPLAY  HELP     LIST     RENAME   USAGE
ATTR     END      HISTORY  MAP      SCRATCH
```

Unambiguous abbreviations are acceptable for all subcommand names except for SCRATCH, which requires at least 3 letters as a safety precaution.

5.8.1 ALIAS

The ALIAS subcommand allows you to assign an alias to a member. Its syntax is:

```
ALIAS   <member> <alias>
```

The named member will be assigned the designated alias name. If the library is a load-module library, the attributes of the the alias will be copied from those of the member, with one exception -- the entry point address of the alias will be assigned according to the rules of the IBM Linkage Editor. If no entry point can be determined by those rules, then the entry point address of the member will be copied.

An alias can only be assigned to a true member name. If an alias name is supplied for <member>, this subcommand will fail.

5.8.2 ATTR

The ATTR subcommand allows you to display the attributes assigned to a member of a load-module library, and, optionally, to alter the values of certain attributes. Its syntax is:

```
ATTR  <member> <attribute-list>
```

The load-module attributes of the named member will be listed. If <attribute-list> does not appear, that is all that is done.

If <attribute-list> does appear, it is a parenthesized list of attributes or their negations, separated by commas and/or blanks. The named attributes will be assigned to the member. For a complete description of load-module attributes refer to the documentation of the IBM Linkage Editor [7]. The permitted attributes are:

```
EXEC   REFR   RENT   REUSE
NOEXEC NOREFR NORENT NOREUSE
```

Notice that these attributes are properties of the directory entry for a member. Therefore, you can alter the attributes of an alias without altering those of the corresponding true name.

5.8.3 DISPLAY

The DISPLAY subcommand lists the directory of a library. Optionally, a range within the directory may be listed. The syntax is:

DISPLAY <start> <end>

If no parameter is given the entire directory is listed. If one parameter is given, the directory is listed from the (virtual) member named to the end of the directory. If two parameters are given, all member names within the range so implied are listed.

5.8.4

The END subcommand terminates the LIBMAINT service. It has no parameters.

5.8.5 HELP

The HELP subcommand provides tutorial information about using the service, and will not be needed by the user who has ready access to this document. it has several options that will not be listed here. The best way to learn its use is to enter the command with no parameters, as:

HELP

HELP will list two additional subcommands, "DO" and "CHANGE", which are not available to the NSW user.

5.8.6 HISTORY

The HISTORY subcommand produces a brief summary of certain update history information kept for load modules. Unlike the attributes listed by ATTR, HISTORY deals with data within the member itself. Therefore, the history of an alias and of its true name are always the same. The syntax is:

HISTORY <member>

5.8.7 LIST

The LIST subcommand lists the contents of a member or alias, so long as it is not a load module. The syntax is:

```
LIST <member>
```

5.8.8 MAP

The MAP subcommand produces a brief reference map of a load module similar to that produced by the MAP option of the linkage editor [7]. Its syntax is:

```
MAP <member>
```

5.8.9 RENAME

The RENAME subcommand renames a member or an alias. Its syntax is:

```
RENAME   <old-name>  <new-name>
```

5.8.10 SCRATCH

The SCRATCH subcommand deletes a member or alias. The member data becomes lost space when no member or alias name remains to refer to it. It is legal to scratch a member name and leave it referenceable only through an alias; however, this can cause problems later (see the ALIAS subcommand), so you are cautioned against it. The syntax is:

```
SCRATCH <member>
```

5.8.11 USAGE

The USAGE subcommand produces a brief listing of the general status of the library. There are no parameters. Included are:

1. The number of directory blocks allocated and in use. When these numbers become nearly equal, you should think of creating a new library with a larger directory.

2. The number of members and aliases in the library.

3. The number of disk tracks allocated and unused. The length of a disk track depends on the physical device type being used. At this writing, NSW data is stored on an IBM 3380 disk pack, with approximately 47,000 characters per track (under optimum conditions). The number of unused tracks does not include space lost due to members deleted or updated.

4. The number of disk extents taken by the library. The "extent" is the vehicle for expanding a library's space allocation. A maximum of sixteen extents are permitted, so when this number approaches sixteen, you should consider compressing or copying the library.

5.9 THE MERGELIB SERVICE

The MERGELIB service copies the members of one library into another. Duplicate names are automatically replaced. Members with aliases are copied only once, but with all aliases retained. The output library must already exist. If it is empty, or freshly garbage-collected and free of duplicate names, then the result of this service will be a garbage-free output library. Thus MERGELIB can be used as a form of library garbage collection procedure. This is especially useful when it is desired to expand the directory as a part of a garbage collection process.

MERGELIB is a foreground service, so its parameters are collected by the CCN Encapsulator process. Thus you cannot end your NSW session until the service has completed.

5.10 THE COMPRESS SERVICE

COMPRESS does garbage collection in a library file. It is a foreground service, so its parameters are collected by the CCN Encapsulator process, and you cannot end your NSW session until it has completed.

5.11 EXAMPLES

```
CCN:  createl
NSW file name:  ?
Enter the NSW file name of the new library,
or ctl-C to back out.
NSW file name:  testl.clib
Enter a GFT -- default is "CARDS"
GFT:  360-
File size in kilochars:  ?
Enter a rough guess of how many thousand bytes of data will
normally be stored in this library, or ctl-C to back out.
File size in kilochars:  1
Maximum members:  ?
enter the maximum number of members you expect to ever
store in this library, or ctl-C to back out.
Maximum members:  10
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.TESTL.CLIB
[now talking to 2ccn (...ccn)]
New library name is $CCN.TESTL.CLIB

CCN:  createl
NSW file name:  testl.llib
Enter a GFT -- default is "CARDS"
GFT:  360-load
File size in kilochars:  3
Maximum members:  10
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.TESTL.LLIB
[now talking to 2ccn (...ccn)]
New library name is $CCN.TESTL.LLIB

CCN:  createl
NSW file name:  testl.plib
Enter a GFT -- default is "CARDS"
GFT:  360-pli-source
File size in kilochars:  2
Maximum members:  5
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.TESTL.PLIB
[now talking to 2ccn (...ccn)]
New library name is $CCN.TESTL.PLIB

CCN:  createl
NSW file name:  testl.plib2
Enter a GFT -- default is "CARDS"
GFT:  360-pli-source
```

```
File size in kilochars:  2
Maximum members:  5
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.TESTL.PLIB2
[now talking to 2ccn (...ccn)]
New library name is $CCN.TESTL.PLIB2

CCN:  putmem
Library filespec:  ?
Enter the NSW file specification of the
  library to which you wish to add
  a member, or control-c to abort.
Library filespec:  ...clib
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.CLIB
[now talking to 2ccn (...ccn)]
Sequential file name:  ?
Enter the NSW file specification of the file
  containing the data which you wish added
  to the library, or control-c to abort.
Sequential file name:  ...lkcntl
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.LKCNTL
[now talking to 2ccn (...ccn)]
member name:  ?
Enter the name under which you wish
  the data stored in the library,
  or control-c to abort.
member name:  lkcntl

CCN:  putmem
Library filespec:  ...plib
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.PLIB
[now talking to 2ccn (...ccn)]
Sequential file name:  ...pgm
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.PGM
[now talking to 2ccn (...ccn)]
member name:  pgm

CCN:  putmem
Library filespec:  ...plib
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.PLIB
[now talking to 2ccn (...ccn)]
Sequential file name:  ...body
```

```
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.BODY
[now talking to 2ccn (...ccn)]
member name:  body

CCN:  getmem
Library filespec:  ?
Enter the NSW file specification of the library
  from which you wish to extract a member,
  or control-c to abort.
Library filespec:  ...clib
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.CLIB
[now talking to 2ccn (...ccn)]
Member name:  ?
Enter the name of the member to be extracted,
  or control-c to abort.
Member name:  lkcntl
Sequential NSW file name:  ?
Enter the NSW entry name of the file to receive
  the library member's data,
  or control-c to abort.
Sequential NSW file name:  getc
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.GETC
[now talking to 2ccn (...ccn)]
NSW filename is $CCN.GETC

CCN:  getmem
Library filespec:  ...plib
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.PLIB
[now talking to 2ccn (...ccn)]
Member name:  pgm
Sequential NSW file name:  getp
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.GETP
[now talking to 2ccn (...ccn)]
NSW filename is $CCN.GETP

CCN:  mergelib
Input filespec:  ?
Enter the NSW file specification of the
  input library, or control-c to abort.
Input filespec:  ...plib
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
```

```
   Full file name is $CCN.TESTL.PLIB
[now talking to 2ccn (...ccn)]
Output filespec:  ?
Enter the NSW file specification of the
  output library, or control-c to abort.
Output filespec:  ...plib2
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.PLIB2
[now talking to 2ccn (...ccn)]

CCN:  libmaint
Filespec:  ?
Enter the NSW file specification of the library
  to be operated on, or control-c to abort.
Filespec:  plib2
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full file name is $CCN.TESTL.PLIB2
[now talking to 2ccn (...ccn)]

PDS:  usage
1 DIRECTORY BLOCKS ALLOCATED
1 DIRECTORY BLOCKS IN USE
2 MEMBERS EXIST
1 TRACKS ALLOCATED
0 TRACKS UNUSED
1 EXTENTS USED
PDS: d
BODY        PGM
PDS:  al pgm newname
ALIAS ASSIGNED
PDS:  scr body
SCRATCHED
PDS:  re pgm renamed
RENAMED
PDS:  di
NEWNAME     RENAMED
PDS:  1 renamed
00000020 TESTOR:  /* SAMPLE PL/I PROGRAM */
00000030   PROCEDURE OPTIONS (MAIN);
00000040     %INCLUDE DECLS;
00000050     %INCLUDE BODY;
00000060   END TESTOR;
PDS: end

CCN:
```

Section 6

GENERAL INFORMATION ON NATIVE-LANGUAGE SUPPORT SERVICES

The CCN IBM Native Language Processing (NLP) service support packages enable you to compile, link, and execute programs written in languages supported by IBM System/360/370. There will be a separate support package in this group for each language supported by System/360/370 and of interest to NSW; at this writing, support is available for PL/I, FORTRAN, and the IBM Assembler. Possible candidates for future support include ALGOL, PASCAL, and COBOL.

This section will introduce you to the general concepts of program development using IBM services under NSW, and to the features common to most NLP support. Subsequent sections give more information on the support for specific languages.

For each language, the same basic set of services is provided. See figure 3 for the relationships among them. They are:

1. A Compile-and-go service
2. A Compile-and-save service
3. A batch Load-and-go service
4. An interactive Load-and-go service
5. A Linkage-editor service

The NLP services are intended to be supported by other CCN services, as illustrated in Figure 2.

The NLP services are mostly batch services, so they expect files as their primary data inputs, and they produce printer files as their primary informational outputs. The EDIT service can be used to prepare input files, and the DISPLAY service can be used to view outputs.

Each support package uses NSW library files to store binary programs, and some also define source-language libraries. This means that each NLP support package must be used in conjunction with the CCN Library Management support services.

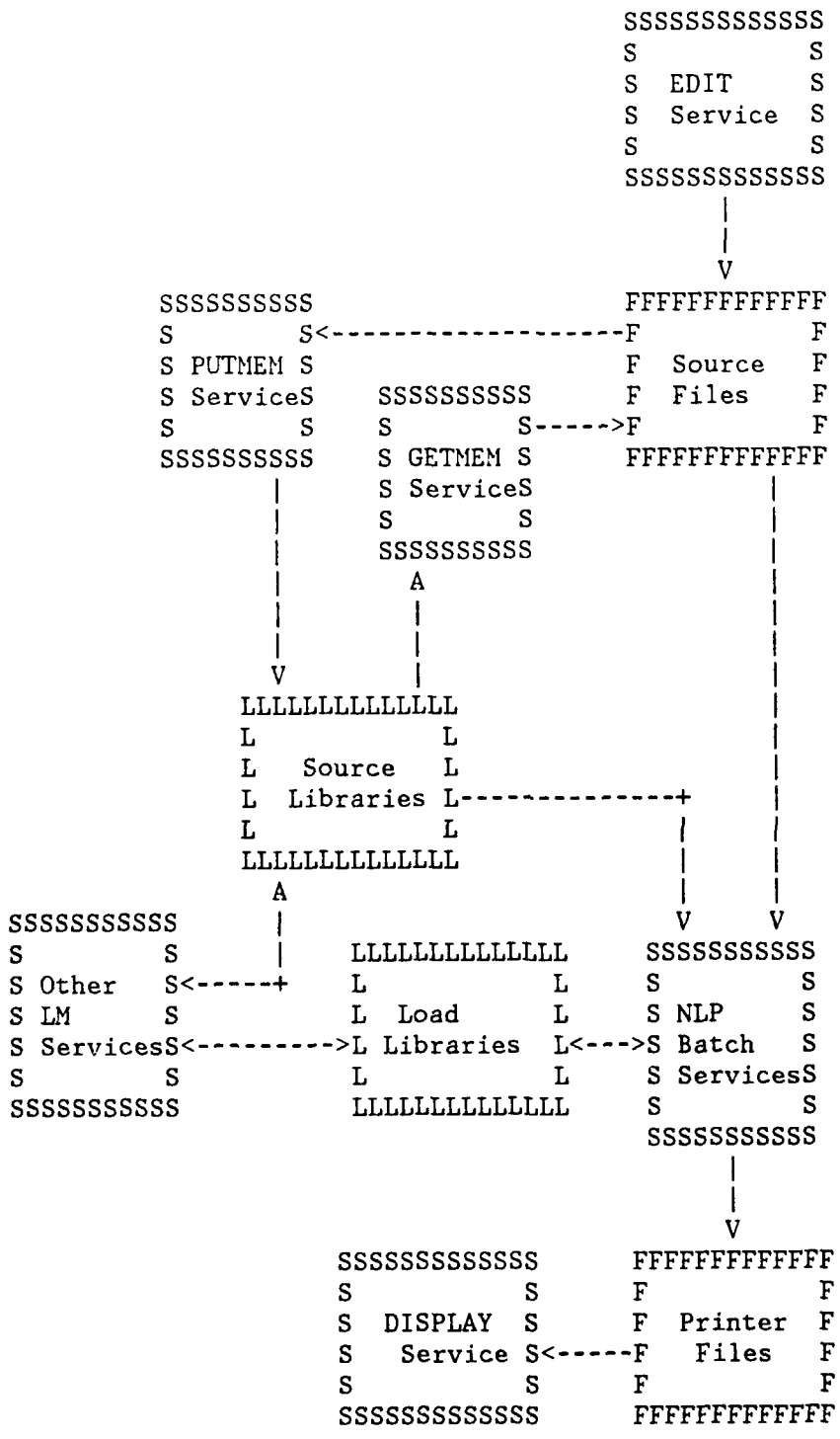


Figure 2: Interfaces Between NLP and Other Services

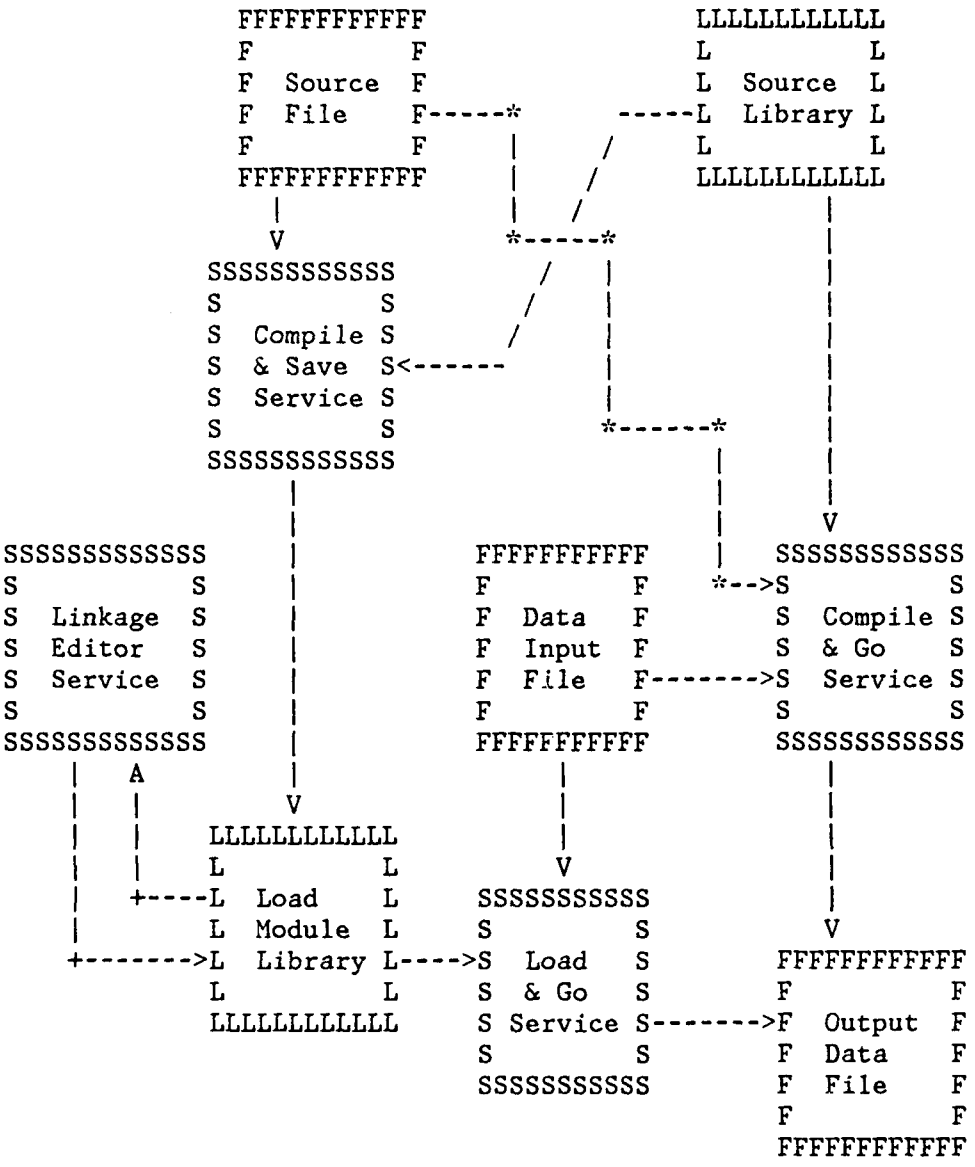


Figure 3: Interfaces Among NLP Services

6.1 COMPILE-AND-GO SERVICES

A compile-and-go service is a batch service that accepts a source program file and compiles it. For the languages that define it, an optional source program library can be specified. The compiled program is not saved, but is executed and discarded. You can supply one or two optional data input files and an optional load-module library to be searched for any separately compiled subroutines called by your main program.

As output, a compile-and-go service produces a printer file containing the compiler listing. If you elect to execute, an execution-time printer listing may be produced. For programs which wish to produce non-printer output, you can specify a data output file. Current restrictions in the NSW Interactive Batch Specifier (IBS) implementation prevent a more comprehensive execution-time file structure at this time.

The standard batch-job system-output listing is also produced.

6.2 COMPILE-AND-SAVE SERVICES

A compile-and-save service is a batch service that accepts a source program file and compiles it, saving the binary output as a designated member of a designated load-module library. For the languages that define it, an optional source program library can be specified. There is no option to execute the program.

As output, a compile-and-go service produces a printer file containing the compiler listing. You must have already created a load-module library, using the CREATEL service, to receive the binary program.

The standard batch-job system-output listing is also produced.

6.3 BATCH LOAD-AND-GO SERVICES

A batch load-and-go service executes a binary program that was produced by a compile-and-save service, or one that has been reprocessed by a linkage-editor service. You specify a member and a library name. The same library will be searched for any separately compiled subroutines called by your main program, if they have not already been bound to it with a linkage-editor service. You can also supply one or two optional data input files.

As output, a batch load-and-go service may produce an execution-time printer listing. For programs which wish to produce non-printer output, you can specify a data output file. Current restrictions in the NSW implementation prevent a more comprehensive execution-time file structure at this time.

The standard batch-job system-output listing is also produced.

6.4 INTERACTIVE LOAD-AND-GO ("IGO") SERVICES

Interactive load-and-go services, collectively referred to as "IGO" services, are more useful than batch load-and-go services, because they allow you to specify a comprehensive set of input and output files.

An IGO service will execute a binary program that was produced by a compile-and-save service, or one that has been reprocessed by a linkage-editor service. You specify a member and a library name. The same library will be searched for any separately compiled subroutines called by your main program, if they have not already been bound to it with a linkage-editor service.

An IGO service executes in three stages: file allocation, execution, and file disposition.

6.4.1 File Allocation

During file allocation, the Encapsulator is driven by your responses to a set of basic prompts. The primary prompt is for the internal name of a file, or for a control-C to indicate that you have allocated all the files that the program will need, and that you are ready to proceed with program execution.

The "internal name" of a file varies from one programming language to another, and the various IGO services will attempt to prompt you in the appropriate parlance. In FORTRAN, you will deal with "logical unit" numbers; in PL/I, "file titles" are used, and Assembler deals directly with the "ddnames" used by the IBM operating system.

Whatever the particular service calls an internal file name, for each one that you specify you will be prompted for several pieces of information. The first of these is the basic type of file allocation to be performed, and the others are selected depending on your response to that first prompt. There are five basic types of allocation:

1. Type "I" -- NSW Input File Allocation

   If you select type "I" allocation, you are prompted for an NSW filespec and a GFT. An unnamed copy of the NSW file is moved into your workspace, converting the data to the stated GFT. This copy is allocated to the internal file. If you enter a null GFT, then no conversion is performed, and the workspace copy will have

the same GFT as the NSW file. In any case, it is your responsibility to ensure that the program and the data are compatible.

Notice that the workspace copy is "unnamed". It lasts only for the duration of the IGO service. If this is not satisfactory for your purposes, consider making the workspace copy with the GET service, and using allocation type "W", described below.

2. Type "O" -- NSW Output File Allocation

If you select type "O" allocation, you are prompted for an NSW filename, a GFT, and an approximate file size. An "unnamed" empty file will be created in the workspace and allocated to the internal file.

In this version of the IGO services, the NSW file named is not looked up until the file disposition phase of the service. This is why there is no option to copy the GFT from the previous generation of the file, should one exist.

Again, if an "unnamed" workspace file is not satisfactory, use allocation type "W", described below, and the DELIVER service.

3. Type "L" -- NSW Library File Allocation

Type "L" allocation is essentially just like type "I" allocation, except that no GFT will be requested. The main difference is that the NSW file is assumed to be a library, so instead of making a workspace copy, the NSW file is allocated to the program directly. This type of allocation usually makes sense only for an Assembler program, as FORTRAN and PL/I do not have the facilities for direct manipulation of libraries.

4. Type "W" -- Workspace File Allocation

Type "W" allocation is quite different from the others because no NSW file is involved. You are prompted for the local name of an existing workspace file, and that file is allocated to the internal file. Since the file must have been previously created, and since workspace files cannot be shared among concurrent users, the service has no need to know the file's GFT or whether it will be read or written. Those things are up to you.

5. Type "*" -- Terminal Allocation

Type "*" allocation is unique in that external files are not involved at all. The internal file is allocated directly to the

user's terminal. This type of allocation can be used with any sequential file provided you do not intend to use the file conversationally, and it can be used for conversational PL/I STREAM files. For other languages, internal buffering of file data will interfere with conversational use. Of course, in Assembler, you have facilities to read and write the terminal without the use of files.

6.4.2 Execution

After you have ended the file allocation phase, an IGO service enters its execution phase. You are prompted for the library and member to be executed. You can abort execution by entering control-C; otherwise the program is loaded and executed.

6.4.3 File Disposition

After the execution phase is completed or aborted, the file disposition phase is performed. For each file allocated in the first phase, the binding to the program is unbound. Temporary files in the workspace are deleted, but not those allocated as type "W".

For each file allocated as type "O", you are asked whether delivery is to be performed. If you elect not to deliver, the temporary file is deleted; otherwise, it is delivered to the NSW file space. If delivery fails, you are given the option of giving the unnamed workspace copy a visible name. If you exercise this option, then you can attempt delivery again using the DELIVER service. Of course, this option is only useful if you are operating under the CCN WSCI.

6.5 LINKAGE-EDITOR SERVICES

A linkage-editor service collects compiled and/or previously edited programs from load-module libraries and combines them to form a single executable program. this program is then stored as a member of a load-module library. You specify an input library name, an output library name, and a file containing linkage-editor control statements.

The service produces a file containing a listing of the control statements, a map of the output module, a cross-reference table, and an error list. The standard batch-job system-output listing is also produced.

The linkage-editor services all use the IBM linkage editor, program 360S-ED-521 [7]. The linkage editor is essentially a language-inde-

pendent program, and these services differ primarily in the selection of the system libraries that are implicitly provided for the resolution of compiler-generated calls to execution-time system routines. For this reason, the bulk of the NSW user documentation on these services is contained in this section, rather than being replicated for each language.

6.5.1 Functions of the Linkage Editor

The primary function of the linkage editor is to combine load modules that reference each other into a single new load module. In addition to modules that you explicitly ask to be combined, the editor will search your input library, as well as an implicitly provided, language-dependent system library, for any modules called but not otherwise included in the output module. This feature, called the "automatic library call" mechanism, enables you to collect many load modules merely by including the main program. The resulting module is intended to be fully resolved and ready to execute. A load-and-go service can process it much faster than it can an ordinary compiled program, so programs that are executed frequently and that do not change should be stored in this form. A single use of a linkage-editor service can create any number of output modules.

An essential, but less often needed, function of the linkage editor is the actual editing of inter-module linkages. That is, the alteration of a module reference to indicate a different module, or the deletion of a portion of an included load module. Another form of editing is the construction of overlay structures of load modules; however, since the CCN system uses virtual memory, this function is not likely to be needed.

Proper use of the linkage-editor services requires an understanding of IBM load-module structures and dynamics that is beyond the scope of NSW documentation. If the control statement documentation below is not clear to you, you should obtain a copy of [7].

6.5.2 Control Statement Usage

A linkage editor control statment consists of a verb, delimited on *both* sides by one or more blanks, followed by an operand field in which no blanks may occur. The sections below define a subset of the available statements. Within this subset, statements are grouped quite informally, with only two rules that need to be followed:

1. Multiple output modules can be created with a single file of statements, but you must group all the statements relevant to a single output module together. The end of such a group is the NAME statement. Until a NAME statement is encountered, the editor is only collecting information. On reaching a NAME, it stops collecting and stores the member. Any subsequent statements define a new output member.

2. The CHANGE and REPLACE statements must immediately precede the INCLUDE of the module to which they are to apply. If the INCLUDE specifies more than one module, only the first one is affected. Again, you can think of the editor as collecting editing instructions until a module is included, then applying them all and forgetting them before proceeding with the next module.

6.5.2.1 INCLUDE -- Getting a Member From a Library

The NSW services support two forms of the INCLUDE statement:

```
INCLUDE SYSLIB(<member>,<member>,...)
INCLUDE SYSLMOD(<member>,<member>,...)
```

The statement specifies one or more members to be included in the current output module. If there are any pending CHANGE or REPLACE operations, they will be applied to the first or only member name. In the first form, your input library is searched for the members. If any are not found, the implicit system library is searched. If any are still not found, an error exists. In the second form, the *output* library (only) is searched for the members. If any are not found, an error exists. You can use as many INCLUDE statements as you need, and you can include as many members on each as you can type without exceeding column 72.

6.5.2.2 ENTRY -- Specifying the Primary Entry Name

The ENTRY statement has the form:

```
ENTRY    <name>
```

The given name must be an entry name within the collected modules by the time the member is actually written. It need not have been found at the time this control statement is processed. For all practical purposes, this statement is always required by the NSW linkage-editor services. It should appear just once per module.

6.5.2.3 NAME -- Naming the Output Module

The NAME statement has two forms:

```
NAME    <name>
NAME    <name>(R)
```

All processing of the current output member is completed, and it is added to the library as the named member. The entry point associated with the module name will be the one specified in the NAME statement. In the first form, if the name already exists in the directory, it will not be replaced. In the second form, it will be.

6.5.2.4 ALIAS -- Assigning Aliases

The ALIAS statement has the form:

```
ALIAS  <name>,<name>,...
```

You can assign up to 16 aliases to the module. You can use as many ALIAS statements as you need, and you can include as many names on each as you can type without exceeding column 72. Each name has an entry point associated with it. This is determined separately for each name, according to these two rules:

1. If the alias name matches an entry name within the module, then that entry will be assigned.

2. If the alias name does not match any entry name within the module, then the entry specified by the ENTRY statement will be assigned.

6.5.2.5 LIBRARY -- Leaving References Unresolved

The LIBRARY statement has two forms:

```
LIBRARY  (<name>,<name>,... )
LIBRARY *(<name>,<name>,... )
```

This statement lists the names of external references for which the automatic library call mechanism is to be disabled. Unless a named entry point was collected as a result of an INCLUDE statement, it will not be included, references to it will remain unresolved, and a warning-level message will be issued.

The second form of LIBRARY has the same meaning, except that the output module will be marked in such a way as to propogate the effect of the LIBRARY statement to any future linkage-edit that includes this output module. This is called the "NEVER-CALL" function.

6.5.2.6 CHANGE -- Renaming Things

The CHANGE statement has the form:

```
CHANGE <name>(<newname>),<name>(<newname>),...
```

You can use as many CHANGE statements as you need, and you can include as many changes on each as you can type without exceeding column 72. All changes are collected, and take effect simultaneously when the next included module is fetched.

The names may be entry names, control section names, external reference names, or any sort of names known to the linkage editor. Each <name> is replaced by the corresponding <newname>, in one module only. If you wish changes to have effect in more than one module, you must replicate the CHANGE cards.

6.5.2.7 REPLACE -- Deleting Things

The REPLACE statement has the form:

```
REPLACE <name>,<name>,...
```

You can use as many REPLACE statements as you need, and you can include as many names on each as you can type without exceeding column 72. All replacements are collected, and take effect simultaneously when the next included module is fetched.

The names may be entry names, control section names, external reference names, or any sort of names known to the linkage editor. Each one is deleted from one module only. If you wish deletions to have effect in more than one module, you must replicate the REPLACE cards.

6.6 EXAMPLES

Because the NSW interaction is so similar for all the NLP packages, we only show one set of representative examples.

```
NSW:  use ...pli [confirm] !
  [command initiated]
  Beginning specification of job for batch tool
           PUBLIC.SERVICES.CCN.PLI
  Time estimate in seconds (default 20): [confirm] !
  Source file: ...pgm [confirm] !
           Input file is CCN.TESTP.PGM
  Will you be using a source library? (Yes or No): y [confirm] !
  Source library name: ...plib [confirm] !
           Input file is CCN.TESTP.PLIB
  Compiler listing file (default PLI.LIST): [confirm] !
           Output file is CCN.TESTP.PLI.LIST
  Will you be using a subroutine library?
                       (Yes or No): n [confirm] !
  Execution SYSPRINT file (default PLI.SYSPRINT): [confirm] !
           Output file is CCN.TESTP.PLI.SYSPRINT
  Do you have a SYSIN file? (Yes or No): y [confirm] !
  SYSIN file name: ...pgm [confirm] !
           Input file is CCN.TESTP.PGM
  Do you have an INPUT file? (Yes or No): n [confirm] !
  Do you have an OUTPUT file? (Yes or No): n [confirm] !
  SYSOUT file (default PLI.SYSOUT): [confirm] !
           Output file is CCN.TESTP.PLI.SYSOUT
  [command completed]
  Job number is 50250
  Specification of batch job complete


NSW: show stat job 50250 [confirm] !
  [command initiated]
  [command completed]
  The status of job number 50250
                for COMPASS + LUDLAM is: Sending
    Job processing begun
    Job allocated on batch host


NSW:
  The status of job number 50250
                for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.TESTP.PGM ready at batch host
    File CCN.TESTP.PLIB ready at batch host
    File CCN.TESTP.PGM ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 6230
    Execution complete per call from batch host
```

```
    File CCN.TESTP.PLI.SYSOUT delivered into NSW
    File CCN.TESTP.PLI.SYSPRINT delivered into NSW
    File CCN.TESTP.PLI.LIST delivered into NSW


NSW:  use ...plicomp [confirm] !
  [command initiated]
  Beginning specification of job for batch tool
           PUBLIC.SERVICES.CCN.PLICOMP
  Time estimate in seconds (default 20): [confirm] !
  Source file: ...pgms2 [confirm] !
           Input file is CCN.TESTP.PGMS2
  Will you be using a source library? (Yes or No): n [confirm] !
  Compiler listing file (default PLICOMP.LIST): [confirm] !
           Output file is CCN.TESTP.PLICOMP.LIST
  Will you save the binary program?
                      (Yes or No): y [confirm] !
  Library to receive binary program: ...llib [confirm] !
           Input file is CCN.TESTP.LLIB
  Binary member name: main [confirm] !
  SYSOUT file (default PLICOMP.SYSOUT): [confirm] !
           Output file is CCN.TESTP.PLICOMP.SYSOUT
  [command completed]
  Job number is 50251
  Specification of batch job complete


NSW:
  The status of job number 50251
               for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.TESTP.PGMS2 ready at batch host
    File CCN.TESTP.LLIB ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 6267
    Execution complete per call from batch host
    File CCN.TESTP.PLICOMP.SYSOUT delivered into NSW
    File CCN.TESTP.PLICOMP.LIST delivered into NSW


NSW:  use ...plibgo [confirm] !
  [command initiated]
  Beginning specification of job for batch tool
           PUBLIC.SERVICES.CCN.PLIBGO
  Time estimate in seconds (default 20): [confirm] !
  Compiled library name: ...llib [confirm] !
           Input file is CCN.TESTP.LLIB
  Member name of main program: main [confirm] !
  Execution SYSPRINT file (default PLIBGO.SYSPRINT): [confirm] !
           Output file is CCN.TESTP.PLIBGO.SYSPRINT
  Do you have a SYSIN file? (Yes or No): n [confirm] !
  Do you have an INPUT file? (Yes or No): y [confirm] !
  Is your INPUT binary? (Yes or No): n [confirm] !
```

```
  INPUT file name: ...pgm [confirm] !
           Input file is CCN.TESTP.PGM
  Do you have an OUTPUT file? (Yes or No): y [confirm] !
  Is your OUTPUT binary? (Yes or No): n [confirm] !
  OUTPUT file name (default PLIBGO.OUTPUT): [confirm] !
           Output file is CCN.TESTP.PLIBGO.OUTPUT
  SYSOUT file (default PLIBGO.SYSOUT): [confirm] !
           Output file is CCN.TESTP.PLIBGO.SYSOUT
  [command completed]
  Job number is 50252
  Specification of batch job complete


NSW:
  The status of job number 50252
                for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.TESTP.LLIB ready at batch host
    File CCN.TESTP.PGM ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 7521
    Execution complete per call from batch host
    File CCN.TESTP.PLIBGO.SYSOUT delivered into NSW
    File CCN.TESTP.PLIBGO.OUTPUT delivered into NSW
    File CCN.TESTP.PLIBGO.SYSPRINT delivered into NSW


NSW:  use ...plilink [confirm] !
  [command initiated]
  Beginning specification of job for batch tool
            PUBLIC.SERVICES.CCN.PLILINK
  Control statement file: ...lkcntl [confirm] !
           Input file is CCN.TESTP.LKCNTL
  Compiled library name: ...llib [confirm] !
           Input file is CCN.TESTP.LLIB
  Module map file (default PLILINK.LIST): [confirm] !
           Output file is CCN.TESTP.PLILINK.LIST
  Library to receive linked program: ...llib [confirm] !
           Input file is CCN.TESTP.LLIB
  SYSOUT file (default PLILINK.SYSOUT): [confirm] !
           Output file is CCN.TESTP.PLILINK.SYSOUT
  [command completed]
  Job number is 50253
  Specification of batch job complete


NSW:
  The status of job number 50253
                for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.TESTP.LKCNTL ready at batch host
    File CCN.TESTP.LLIB ready at batch host
```

```
    File CCN.TESTP.LLIB ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 7521
    Execution complete per call from batch host
    File CCN.TESTP.PLILINK.SYSOUT delivered into NSW
    File CCN.TESTP.PLILINK.LIST delivered into NSW
    File CCN.TESTP.PLILINK.SYSPRINT delivered into NSW


NSW:  use ...ccn [confirm] !
  [command initiated]
     .
     .
     .
CCN:  pliigo
First file title or ctl-C:  ?
 Enter a file title to be defined.
 Control-C ends the definitions.
First file title or ctl-C:  sysprint
Type:  I, O, L W, or *:  ?
 What kind of file is to be allocated to this file title?
 Enter "I" if to an NSW Input file;
 Enter "O" if to an NSW Output file;
 Enter "L" if to an existing NSW Library file;
 Enter "W" if to an existing local workspace file;
 Enter "*" if to the terminal;
 Carriage return defaults to "I";
 Or enter ctl-C to forget this file title.
Type:  I, O, L W, or *:  O
NSW filename:  ?
 Enter the filename of the NSW file to receive the data from
 output file sysprint, or ctl-C to back out.
NSW filename:  pliigo.sysprint
Enter a GFT -- default is "LIST"
GFT:  360-print
File size in kilochars: ?
 Enter a rough guess of how many thousand bytes of data will
 be written to this file, or ctl-C to back out.
File size in kilochars: 2
NSW file pliigo.sysprint allocated to file title sysprint
    as OUTPUT type 360-print.

Next file title or ctl-C:  input
Type:  I, O, L W, or *:  *
file title input allocated to terminal

Next file title or ctl-C:  output
Type:  I, O, L W, or *:  *
file title output allocated to terminal

Next file title or ctl-C:  @
3 files allocated, including 1 potential deliverables.

Compiled library name:  ?
```

```
 Enter the filespec of the NSW Library file that contains the
 programs to be run, or enter ctl-C to abort.
Compiled library name:  ...llib
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
    Full file name is $CCN.TESTP.LLIB
[now talking to 2ccn (...ccn)]
Main Program name:  ?
 Enter the library member name of the compiled program that
 is to be the main entry point, or enter ctl-C to abort.
Main Program name:  echoer
Do you want a memory map?  ?
 Yes" will map the loaded program on your terminal before
 it executes.  "No" will suppress the map.  Ctl-C will abort.
Do you want a memory map?  no
Enter any "parm" string to be passed the program:  ?
 Enter that character string that the main program expects
 to receive from the system.  A null string is legitimate.
 Ctl-C will abort.
Enter any "parm" string to be passed the program:

PL/I ECHO TEST PROGRAM...
ENTER A STRING:  this is user input
ECHO OF STRING:  this is user input
ENTER A STRING:  a second line...
ECHO OF STRING:  a second line...
ENTER A STRING:
END OF ECHO TEST

Should file title sysprint be delivered
   to NSW file pliigo.sysprint? y
[...Disconnecting from 2ccn]
[NSWExec for 2ccn:]
   Full name of new file is
     $CCN.TESTP.PLIIGO.SYSPRINT
[now talking to 2ccn (...ccn)]
File title sysprint delivered to NSW file pliigo.sysprint.

CCN:
```

Section 7

FORTRAN SUPPORT

The CCN FORTRAN support package enables you to compile, link, and execute programs written in the FORTRAN language, using the IBM "G1" compiler and library (IBM program products 5743-FO2 and 5743-LM3) and the IBM Linkage Editor (IBM program 360S-ED-521). This section describes the features of the FORTRAN support package that are unique among the CCN Native Language Processing (NLP) support packages. The general concepts of such packages are described in an earlier section. The specific features of the IBM FORTRAN compiler and Linkage Editor are documented in IBM publications. The package uses NSW library files to store binary programs, so it must be used in conjunction with the CCN Library Management (LM) support package. Most of the services are batch services, so they produce printer files as their primary informational outputs. The DISPLAY service should be used to view such outputs.

In addition to the other sections of this manual, the user of this support package may need the following additional documentation:

The IBM FORTRAN language specifications [9]
The IBM FORTRAN programmer's guide [10]
The IBM Linkage Editor manual [7]
The IBM Linkage Editor messages manual [8]

The FORTRAN support package contains the same basic set of services as all NLP support packages. They are:

FORTRAN, a compile-and-go service
FORTCOMP, a compile-and-save service
FORTBGO, a batch load-and-go service
FORTIGO, an interactive load-and-go service
FORTLINK, a Linkage-editor service

7.1 FORTRAN: COMPILE-AND-GO

FORTRAN is a batch service that compiles and executes a FORTRAN source program. The compiled program is not saved. You can specify from one to four execute-time files, as explained below under the FORTBGO service. The standard batch-job system-output listing (SYSOUT) is also produced.

7.2 FORTCOMP: COMPILE-AND-SAVE

FORTCOMP is a batch service that accepts a source program file and compiles it, optionally saving the binary output as a designated member of a designated load-module library. There is no option to execute the program.

As output, FORTCOMP produces a printer file containing the compiler listing. If you save the binary program, you must have already created a load-module library, using CREATEL, to receive it. The standard batch-job system-output listing (SYSOUT) is also produced.

7.3 FORTBGO: BATCH LOAD-AND-GO

FORTBGO executes a binary program that was produced by FORTCOMP or FORTLINK. You specify a member and a library name. The same library will be searched for any separately compiled subroutines called by your main program, if they have not already been bound to it with FORTLINK. You have the same options for execute-time files as under the FORTRAN service:

1. Logical Unit 5

   Logical unit 5 is the default unit for reading the primary input file. It is not required, but if you use it, it will be of type 360-CARDS.

2. Logical Unit 6

   A file must always be assocaited with logical unit 6, since the FORTRAN library routines use it. It is also available to the programmer as an output file of type 360-PRINT.

3. Logical Unit 15

   Logical unit 15 has been chosen as an optional second input file. Its GFT will depend on how you respond, when specifiying the service parameters, to questions about the class of the file. If

you request a text file, it will be 360-LIST, which is suitable for reading with FORTRAN formatted I/O statements. If you request a binary file, it will be 360-BINARY, which is suitable for reading with FORTRAN unformatted I/O statements.

4. Logical Unit 16

   Logical unit 16 has been chosen as an optional second output file. Its GFT will depend on how you respond, when specifiying the service parameters, to questions about the class of the file. If you request a text file, it will be 360-LIST, which is suitable for writing with FORTRAN formatted I/O statements. If you request a binary file, it will be 360-BINARY, which is suitable for writing with FORTRAN unformatted I/O statements.

7.4 FORTIGO: INTERACTIVE LOAD-AND-GO

FORTIGO loads a compiled FORTRAN program and executes it in the foreground. It follows the conventions outlined for the "IGO" services in the section on Native-Language Support Packages.

FORTIGO refers to internal files as "logical units", as is appropriate in a FORTRAN environment. When you type a logical unit number, you must express it as *two numeric digits*. The service does not check for this form, and if you violate it, the program will be unable to use the file.

There is currently no support for the form of file allocation required by the continued use of a logical unit after the execution of the FORTRAN END FILE statement against it.

FORTRAN files allocated as type "*" will be fully buffered. This almost precludes the use of terminal input files, but it is of little consequence in the case of simple output files.

7.5 FORTLINK: PERFORM LINKAGE EDITING

FORTLINK uses the IBM Linkage Editor to collect sets of binary programs output by FORTCOMP and FORTLINK (as well as those produced by the Assembler services) and to store them, as monolithic executable programs, as members of a load-module library. You specify an input library name, an output library name, and a file containing linkage-editor control statements. Note that, since this service can store more than one member into the output library, the member names are specified as a part of the control file, not as a part of the service parameters.

The service produces a file containing a listing of the control statements, maps of the output modules, cross-reference tables, and

error lists. The standard batch-job system-output listing (SYSOUT) is also produced.

The functions of the linkage editor are summarized in the section on Native Language Support.

7.6 PREPARING FORTRAN SOURCE FILES

The primary source input file for FORTRAN and FORTCOMP is a FORTRAN-SOURCE file. In order to ensure that conversion to this type does not alter your data, you must prepare it with these restrictions in mind:

1. The maximum length of any data line (not including any sequence number fields) is 72 characters. Any lines longer than this will be split after the 92nd character. Because of the peculiar continuation convention used by FORTRAN, this will alter the meaning of the program, and it will usually produce compilation errors.

2. If the file has sequence numbers, they must be declared to NSW. That is, the file must be given a GFT the attributes of which includes a sequence field (such as FORTRAN-SOURCE). Sequence numbers will occupy columns 73-80.

3. If the file has no sequence numbers, then columns 73-80 will be blank.

4. All characters of the file will be forced to upper case.

5. Binary files cannot be used.

Section 8

PL/I SUPPORT

The CCN PL/I support package enables you to compile, link, and execute programs written in the PL/I language, using the IBM Optimizing compiler and subroutine library (IBM program product composite package S734-PL3) and the IBM Linkage Editor (IBM program 360S-ED-521). This document describes the features of the PL/I support package that are unique among the CCN Native Language Processing (NLP) support packages. The general concepts of such packages are described in another section. The specific features of the IBM PL/I optimizing compiler and Linkage Editor are documented in IBM publications. The package uses NSW library files to store binary programs and source text segments, so it must be used in conjunction with the CCN Library Management (LM) support package. Most of the services are batch services, so they produce printer files as their primary informational outputs. The DISPLAY service should be used to view such outputs.

In addition to the other sections of this manual, the user of this support package may need the following additional documentation:

The IBM PL/I language specifications [11]
The IBM PL/I programmer's guide [12]
The IBM PL/I TSO user's guide [13]
The IBM PL/I messages compendium [14]
The IBM Linkage Editor manual [7]
The IBM Linkage Editor messages manual [8]

The PL/I support package contains the same basic set of services as all NLP support packages. They are:

PLI, a compile-and-go service
PLICOMP, a compile-and-save service
PLIBGO, a batch load-and-go service
PLIIGO, an interactive load-and-go service
PLILINK, a Linkage-editor service

8.1 PLI: COMPILE-AND-GO

PLI is a batch service that compiles and executes a PL/I source program, optionally using a source library. The compiled program is not saved. You can specify from one to four execute-time files, as explained below under the PLIBGO service. The standard batch-job system-output listing (SYSOUT) is also produced.

8.2 PLICOMP: COMPILE-AND-SAVE

PLICOMP is a batch service that accepts a source program file and compiles it, optionally saving the binary output as a designated member of a designated load-module library. An optional source program library can be specified. There is no option to execute the program.

As output, PLICOMP produces a printer file containing the compiler listing. If you save the binary program, you must have already created a load-module library, using CREATEL, to receive it. The standard batch-job system-output listing (SYSOUT) is also produced.

8.3 PLIBGO: LOAD-AND-GO IN BATCH

PLIBGO executes a binary program that was produced by PLICOMP or PLILINK. You specify a member and a library name. The same library will be searched for any separately compiled subroutines called by your main program, if they have not already been bound to it with PLILINK. You will also be prompted for various execute-time files.

PLIBGO allows the use of only a limited set of execute-time files. Your PL/I program can use any subset of them. Current limitations on the type of information that can be specified as service parameters prevent the use of arbitrary file titles and types. These limitations are expected to be lifted in a future NSW release. In the meantime you are limited to a fixed set of file titles with fixed GFT's.

1. SYSIN

   SYSIN is the default title for the major STREAM INPUT file; however, by explicitly declaring it, you can make it into a RECORD file. It is not required, but if you use it, it must be of type 360-CARDS.

2. SYSPRINT

   The file titled SYSPRINT is always required, since the PL/I library routines use it. It is also available to the programmer. It is a STREAM OUTPUT file of type 360-PRINT.

3. INPUT

   INPUT is the title chosen for the optional second INPUT file. It can be STREAM or RECORD, depending on how you declare it. Its GFT will depend on how you respond, when specifiying the service parameters, to questions about the class of the file. If you request a text file, it will be 360-LIST. If you request a binary file, it will be 360-BINARY.

4. OUTPUT

   OUTPUT is the title chosen for the optional second OUTPUT file. It can be STREAM or RECORD, depending on how you declare it. Its GFT will depend on how you respond, when specifiying the service parameters, to questions about the class of the file. If you request a text file, it will be 360-LIST. If you request a binary file, it will be 360-BINARY.

8.4 PLIIGO: LOAD-AND-GO INTERACTIVELY

PLIIGO loads a compiled PL/I program and executes it in the foreground. It follows the conventions outlined for the "IGO" services in the section on Native-Language Support Packages.

PLIIGO refers to internal files as "file titles", as is appropriate in a PL/I environment. The file title is either the name specified in the TITLE option of the OPEN statement, or the name of the FILE datum

PL/I STREAM files can be allocated to your terminal by using allocation type "*". The PL/I run-time package will then translate file-management requests into terminal-management requests. This eliminates record buffering, and enables input and output files to be used together to achieve true conversational terminal management.

If PL/I RECORD files are allocated as type "*", buffering will still take place, and conversational use will not be possible.

8.5 PLILINK: PERFORM LINKAGE EDITING

PLILINK uses the IBM Linkage Editor to collect sets of binary programs output by PLICOMP and PLILINK (as well as those produced by the Assembler services) and to store them, as monolithic executable programs, as members of a load-module library. You specify an input library name, an output library name, and a file containing linkage-editor control statements. Note that, since this service can store more than one member into the output library, the member names are specified as a part of the control file, not as a part of the service parameters.

The service produces a file containing a listing of the control statements, maps of the output modules, cross-reference tables, and error lists. The standard batch-job system-output listing (SYSOUT) is also produced.

The functions of the linkage editor are summarized in the section on Native Language support.

8.6 PREPARING SOURCE FILES

The primary source input file for PLI and PLICOMP is a PLI-SOURCE file. If you elect to supply a source library to either of these services, it should be a PLI-SOURCE library. In order to ensure that conversion to this type does not alter your data, you must prepare it with these restrictions in mind:

1. The maximum length of any data line (not including any sequence number fields) is 92 characters. Any lines longer than this will be split after the 92nd character.

2. If the file has sequence numbers, they must be declared to NSW. That is, the file must be given a GFT the attributes of which includes a sequence field, such as PLI-SOURCE. Sequence numbers will occupy the eight columns preceding the first data column. If the file has no sequence numbers, then the eight columns preceding the first data column will be blank.

3. Page-defining format effectors are not supported. If you use them, the results may not be what you expected.

4. The "ASA carriage control" option of the IBM PL/I compiler is not automatically supported. You can enable it with a "*PROCESS" statement [11]; however, if any input lines are split to accomodate the 92-character limit, a program character will probably fall into the carriage-control column. We reccommend that you use the "%SKIP" and "%PAGE" statements instead of ASA control.

5. Binary files cannot be used.

Section 9

IBM ASSEMBLER SUPPORT

The CCN IBM Assembler support package enables you to compile, link, and execute programs written in the System/360/370 Assembler language, using the IBM H-level assembler (IBM program product 5734-AS1) and the IBM Linkage Editor (IBM program 360S-ED-521). This document describes the features of the Assembler support package that are unique among the CCN Native Language Processing (NLP) support packages. The general concepts of such packages are described in a previous section. The specific features of the IBM Assembler and Linkage Editor are documented in IBM publications.

The package uses NSW library files to store binary programs and source text segments, so it must be used in conjunction with the CCN Library Management (LM) support package. Most of the services are batch services, so they produce printer files as their primary informational outputs. The DISPLAY service should be used to view such outputs.

In addition to the other sections of this manual, the user of this support package may need the following additional documentation:

The IBM SYSTEM/370 architecture description [15]
The IBM general Assembler language specifications [16]
The IBM Assembler H language specifications [17]
The IBM Assembler H programmer's guide [18]
The IBM Assembler H messages compendium [19]
The IBM Linkage Editor manual [7]
The IBM Linkage Editor messages manual [8]
The IBM Supervisor services and macros manual [20]
The IBM data-management services manual [21]
The IBM data-management macros manual [22]
The IBM TSO command-processor manual [23]

The Assembler support package contains the same basic set of services as all NLP support packages. They are:

ASM, a compile-and-go service
ASMCOMP, a compile-and-save service
ASMBGO, a batch load-and-go service
ASMIGO, an interactive load-and-go service
ASMLINK, a Linkage-editor service

9.1 ASM: COMPILE-AND-GO

ASM is a batch service that assembles and executes an Assembler source program, optionally using a source library. The compiled program is not saved. You can specify from one to four execute-time files, as explained below under the ASMBGO service. The standard batch-job system-output listing (SYSOUT) is also produced.

9.2 ASMCOMP: COMPILE-AND-SAVE

ASMCOMP is a batch service that accepts a source program file and compiles it, optionally saving the binary output as a designated member of a designated load-module library. An optional source program library can be specified. There is no option to execute the program.

As output, ASMCOMP produces a printer file containing the compiler listing. If you save the binary program, you must have already created a load-module library, using CREATEL, to receive it. The standard batch-job system-output listing (SYSOUT) is also produced.

9.3 ASMBGO: LOAD-AND-GO

ASMBGO executes a binary program that was produced by ASMCOMP or ASMLINK. You specify a member and a library name. The same library will be searched for any separately compiled subroutines called by your main program, if they have not already been bound to it with ASMLINK. You will also be prompted for several execute-time files.

ASMBGO allows the use of only a limited set of execute-time files. Your Assembler program can use any subset of them. Current limitations on the type of information that can be specified as service parameters prevent the use of arbitrary file names and types. These limitations are expected to be lifted in a future NSW release. In the meantime you are limited to a fixed set of DDNAMEs with fixed GFT's.

1. SYSIN

   SYSIN is a DDNAME associated with a text input file of type 360-CARDS.

2. SYSPRINT

   SYSPRINT is a DDNAME associated with a text output file of type 360-PRINT.

3. INPUT

INPUT is the DDNAME chosen for the optional second INPUT file. If you request a text file, it will be 360-LIST. If you request a binary file, it will be 360-BINARY.

4. OUTPUT

OUTPUT is the DDNAME chosen for the optional second OUTPUT file. If you request a text file, it will be 360-LIST. If you request a binary file, it will be 360-BINARY.

9.4 ASMIGO: LOAD-AND-GO INTERACTIVELY

ASMIGO loads an assembled program and executes it in the foreground. It follows the conventions outlined for the "IGO" services in the section on Native-Language Support Packages.

ASMIGO refers to internal files as "ddnames", as is appropriate in an assembler environement.

QSAM and BSAM files can be allocated as type "*"; however, the proper way to achieve true conversational terminal use in Assembler programs is through the GETLINE, PUTLINE, and PUTGET macros [23]. TGET and TPUT [23] can also be used, but these macros bypass many of the TSO session-management facilities.

9.5 ASMLINK: PERFORM LINKAGE EDITING

ASMLINK uses the IBM Linkage Editor to collect sets of binary programs output by ASMCOMP and ASMLINK (as well as those produced by other services). and to store them, as monolithic executable programs, as members of a load-module library. You specify an input library name, an output library name, and a file containing linkage-editor control statements. Note that, since this service can store more than one member into the output library, the member names are specified as a part of the control file, not as a part of the service parameters. The service produces a file containing a listing of the control statements, maps of the output modules, cross-reference tables, and error lists. The standard batch-job system-output listing (SYSOUT) is also produced.

The functions of the linkage editor are summarized in the section on Native Language support.

9.6 PREPARING SOURCE FILES

The primary source input file for ASM and ASMCOMP is a ASM-SOURCE file. If you elect to supply a source library to either of these services, it should be a ASM-SOURCE library, or at least some form of card-image library. In order to ensure that conversion to this type does not alter your data, you must prepare it with these restrictions in mind:

1. The maximum length of any data line (not including any sequence number fields) is 72 characters. Any lines longer than this will be split after the 72nd character, causing invalid assembler statements.

2. If the file has sequence numbers, they must be declared to NSW. That is, the file must be given a GFT the attributes of which includes a sequence field, such as ASM-SOURCE. Sequence numbers will occupy columns 73 through 80.

3. If the file has no sequence numbers, then columns 73-80 will be blank.

4. All characters of the file will be forced to upper case.

5. Binary files cannot be used.

Section 10

MISCELLANEOUS SERVICES

10.1 THE HELP SERVICE

The HELP service provides function, syntax, and operand information on the interactive and batch services available at CCN. HELP itself is an interactive CCN service.

After calling up HELP from the CCN WSCI or the USE command, you are prompted for a "help spec". The simplest such spec is a null string, which requests a summary of CCN services. Another useful spec is a simple service name, which requests a summary of the purpose and use of the named service. Other forms of the help spec can be used to list more specific or more detailed information. The complete format can be listed by entering "HELP" as the help spec. You can abort any segment of HELP output at any time by striking "control-C".

10.2 THE GPSS SERVICE

10.2.1 Functional Description

GPSS (General Purpose Simulation System) [4, 5] is a program designed for conducting evaluations and experiments of systems, methods, and designs through computer simulation. The version of GPSS available at CCN is GPSS V. This is IBM's program product version, compatible with GPSS/360, the older version, but which includes several enhancements.

When used through NSW, some GPSS features are not available, because the program is constrained to reading a single sequential input file, and to producing a single sequential output listing.

10.2.2 User Prompts

After you type the "USE" command that designates the GPSS service, you will be prompted for several necessary pieces of information:

1. Time Estimate in Seconds

   As always, estimating execution time is guesswork at best. The default of 20 seconds should be more than enough for most test runs.

2. Input File

   The input file is that NSW file into which you have already placed your GPSS directive data. This data should be formatted according to [4] and [5].

3. Output File

   As the Output File, you should name a file into which the primary GPSS output listing is to be placed. This file need not already exist -- if it does, you will be asked if it should be replaced.

4. SYSOUT File

   The SYSOUT file is that file of system messages that you will want to consult only if you believe that the service executed improperly, or possibly to obtain feedback on execution time. Like the primary output file, this one need not already exist -- if it does, you will be asked if it should be replaced.

10.2.3 Example

```
NSW: use ...gpss [confirm] !
  [command initiated]
  Beginning specification of job for batch tool PUBLIC.SERVICES.GPSS
  Time estimate in seconds (default 20):  [confirm] !
  Input file: ...gpss1.in [confirm] !
          Input file is CCN.GPSS1.IN
  Output file (default GPSS.OUTPUT): gpss.test.output [confirm] !
          Output file is CCN.GPSS.TEST.OUTPUT
  SYSOUT file (default GPSS.SYSOUT): gpss.test.sysout [confirm] !
  [command completed]
          Output file is CCN.GPSS.TEST.SYSOUT
  Job number is 50212
  Specification of batch job complete

NSW: show status job 50212 [confirm] !
  [command initiated]
  [command completed]
  The status of job number 50212 for COMPASS + LUDLAM is: Submitted
    Job processing begun
    Job allocated on batch host
    File CCN.GPSS1.IN ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 5332

NSW:
  The status of job number 50212 for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.GPSS1.IN ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 5332
    Execution complete per call to batch host
    File CCN.GPSS.TEST.SYSOUT delivered
    File CCN.GPSS.TEST.OUTPUT delivered
```

10.3 THE SPSS SERVICE

10.3.1 Functional Description

SPSS, or Statistical Package for the Social Sciences [35], is a system of computer programs for the statistical analysis of social-science data. The package is distributed by National Opinion Research Center.

10.3.1.1 Supported Facilities

One or more statistical procedures can be requested in a single SPSS service call. The following statistical procedures are available:

CONDESCRIPTIVE: Descriptive distributional statistics.

FREQUENCIES: Frequency tables, histograms, and descriptive distributional statistics.

AGGREGATE: Descriptive distributional statistics for aggregated data files.

CROSSTABS: Contingency tables and related measures of association.

BREAKDOWN: Descriptive distributional statistics of a dependent variable among subgroups of the cases in a file.

T-TEST: Pairwise or between group comparison of sample means.

PEARSON CORR: Pearson product-moment correlation coefficients.

NONPAR CORR: Spearman and/or Kendall rank-order correlation coefficients.

SCATTERGRAM: Scattergram of data points and simple linear regression.

PARTIAL CORR: Partial correlation analysis.

REGRESSION: Multiple linear regression using a stepwise forward algorithm. Variable forcing allowed.

ANOVA: Univariate general linear model for analysis of variance and covariance. Main class model, hierchical model, or complete general-linear model available. Multiple classification analysis option.

ONEWAY: Tests for trends, a priori contrasts, a

posteriori contrasts, and one-way analysis of variance.

DISCRIMINAT: Discriminant analysis using a direct or a stepwise forward with backword glance algorithm.

FACTOR: Principal component factoring, principal factor factoring with iteration, Rao's canonical factoring, alpha factoring, or image factoring. Varimax, quartimax, equimax, or oblique (with delta) rotation.

CANCORR: Canonical correlation analysis.

GUTTMAN SCALE: Guttman scaling with the Goodenough technique.

10.3.1.2 Limitations of the NSW SPSS Service

File maintenance facilities are not supported in the present SPSS tool configuration. Only those data manipulation facilities not utilizing files are supported. The following is a list of facilities most likely to be affected.

- Recode or recompute variables.
- Generate new variables within and across cases.
- Declare missing values for variables.
- Subset a file by defining subfiles.
- Select cases for processing by sampling or by logical tests.
- Weight cases.
- Randomly sample cases.
- Label variables and their values on printed output.
- Process numeric and alphanumeric data.
- Save documented files for further use.
- Generate files to be used by other programs.
- Check SPSS control cards before doing production runs.
- Use files generated by the OSIRIS statistical package.

10.3.2 User Prompts

After you type the "USE" command that designates the SPSS service, you will be prompted for several necessary pieces of information:

1. Time Estimate in Seconds

   As always, estimating execution time is guesswork at best. The default of 20 seconds should be more than enough for most test runs.

2. Input File

   The input file is that NSW file into which you have already placed your SPSS directive data. This data should be formatted according to [35].

3. Output File

   As the Output File, you should name a file into which the primary SPSS output listing is to be placed. This file need not already exist -- if it does, you will be asked if it should be replaced.

4. SYSOUT File

   The SYSOUT file is that file of system messages that you will want to consult only if you believe that the service executed improperly, or possibly to obtain feedback on execution time. Like the primary output file, this one need not already exist -- if it does, you will be asked if it should be replaced.

10.3.3 Example

```
NSW: use ...spss [confirm] !
  [command initiated]
  Beginning specification of job for batch tool PUBLIC.SERVICES.SPSS
  Time estimate in seconds (default 20):  [confirm] !
  Input file:  spss1.in [confirm] !
         Input file is CCN.SPSS1.IN
  Output file (default SPSS.OUTPUT): spss.test.output [confirm] !
         Output file is CCN.SPSS.TEST.OUTPUT
  SYSOUT file (default SPSS.SYSOUT): pss.test.sysout [confirm] !
  [command completed]
         Output file is CCN.PSS.TEST.SYSOUT
  Job number is 50213
  Specification of batch job complete

NSW: show status job 50213 [confirm] !
  [command initiated]
  [command completed]
  The status of job number 50213 for COMPASS + LUDLAM is: Submitting
    Job processing begun
    Job allocated on batch host
    File CCN.SPSS1.IN ready at batch host

  The status of job number 50213 for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.SPSS1.IN ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 5469
    Execution complete per call to batch host
    File CCN.PSS.TEST.SYSOUT delivered
    File CCN.SPSS.TEST.OUTPUT delivered
```

10.4 THE ECAP SERVICE

10.4.1 Functional Description

ECAP (Electronic Circuit Analysis Program) [24, 25] is a system of programs designed to aid the electrical engineer in the design and analysis of electronic circuits. ECAP can produce DC, AC, and/or transient analysis of electrical networks.

10.4.2 User Prompts

After you type the "USE" command that designates the ECAP service, you will be prompted for several necessary pieces of information:

1. Time Estimate in Seconds

   As always, estimating execution time is guesswork at best. The default of 20 seconds should be more than enough for most test runs.

2. Input File

   The input file is that NSW file into which you have already placed your ECAP directive data. This data should be formatted according to [24] and [25].

3. Output File

   As the Output File, you should name a file into which the primary ECAP output listing is to be placed. This file need not already exist -- if it does, you will be asked if it should be replaced.

4. SYSOUT File

   The SYSOUT file is that file of system messages that you will want to consult only if you believe that the service executed improperly, or possibly to obtain feedback on execution time. Like the primary output file, this one need not already exist -- if it does, you will be asked if it should be replaced.

10.5 THE EISPACK SUBROUTINE PACKAGE

10.5.1 Functional Description

EISPACK [34] is a powerful eigensystem subroutine package developed at Argonne National Laboratory. It consists of a series of FORTRAN and Assembler subroutines which may be called from a FORTRAN program. EISPACK can be used to compute some or all of the eigenvalues, with or without eigenvectors, of the following types of matrices: complex general, complex hermitian, real general, real symmetric, real symmetric tridiagonal, and certain real nonsymmetric tridiagonal.

EISPACK subroutines must be called from a driver FORTRAN program, thus, the matrix whose eigensystem is being computed may itself be the result of other computations in your program, and the computed eigensystem may conveniently be used in further calculations in your program.

When a call to Eispack is made, the EISPACK monitor selects the sequence of routines in the eigensystem package which will solve your problem as rapidly as possible with reasonable assurance of stability in the calculations.

10.5.2 Accessing EISPACK

EISPACK is a subroutine library, not an NSW service. It is used through the FORTRAN support package documented elsewhere in this manual.

You do not need to do anything special to make EISPACK routines available to your FORTRAN program. Any routine from the following list which is called from a program processed by one of the FORTRAN support services will automatically be included in the executable program.

The following subroutines are in the NSW EISPACK library:

BAKVEC	BALANC	BALBAK	BANDR	BANDV	BISECT
BQR	CBABK2	CBAL	CG	CH	CINVIT
COMBAK	COMHES	COMLR	COMLR2	COMQR	COMQR2
CORTB	CORTH	DGNHEP	DGSHEP	DSNHEP	EGNHEP
EGSHEP	EISPAC	ELMBAK	ELMHES	ELTRAN	ERMSEP
ESNHEP	FIGI	FIGI2	HQR	HQR2	HTRIBK
HTRIB3	HTRIDI	HTRID3	IBCMEP	IMTQLV	IMTQL1
IMTQL2	INTREP	INVIT	MINFIT	ORTBAK	ORTHES
ORTRAN	QZHES	QZIT	QZVAL	QZVEC	RATQR
REBAK	REBAKB	REDUC	REDUC2	RG	RGG
RS	RSB	RSG	RSGAB	RSGBA	RSP
RST	RT	SUPVEP	SVD	TINVIT	TQLRAT
TQL1	TQL2	TRBAK1	TRBAK3	TRED1	TRED2
TRED3	TRIDIB	TSTURM			

Section 11

SUPPORT FOR THE GIM-II INFORMATION SYSTEM

11.1 FUNCTIONAL DESCRIPTION OF GIM-II

TRW's Generalized Information Management (GIM-II) System [36, 37,38,39,40,41,42] is a software program designed to simplify the definition, creation, maintenance and interrogation of a data base, i.e., a collection of files of information. The GIM-II software functions provide for such features as:

1. File Definition
2. Initial Data Base Construction
3. Update, Selection and Retrieval of an item or an extended item
4. Checkpoint and Restoration of a data base
5. Security
6. Data Base Validation and Statistics
7. Recording Transactions on History Files

A number of other techniques and capabilities for field, file and system manipulation and maintenance are also available.

The GIM-II system was prompted by TRW research which indicated a need to replace dependence on periodic printed reports by timely access to specific relevant information. GIM-II was designed to realize these objectives of a generalized data base management system:

1. Flexible data structures suitable to each application (hierarchy - network)
2. A variety of access/search methods
3. Centralized control of the physical organization of data
4. Storage of data in relation to access frequency and response requirements
5. Data independence of programs and devices

6. Integrity of the data base against destruction and/or security breach

7. Recovery and restart techniques in the event of hardware/ software failures

8. User interaction with the data base via inquiry system or language

9. Multiple update/retrieval of information from the data base

GIM-II is an online system; the data base and program libraries all reside on online direct-access storage. Access to the system can be in either:

1. Interactive mode -- where the user communicates with the system from an interactive terminal

2. Batch mode -- where the primary input and output streams are files.

GIM-II has its own Executive and Data-Management system, all operating under the IBM MVS operating system. A batch-mode GIM-II job executes a private instance of the GIM-II executive, while all interactive GIM-II sessions share a single instance of the executive which operates as an essentially never-ending job under MVS. Concurrent instances of the executive can share data bases.

11.2 GIM-II UNDER NSW

The GIM-II NSW services are specially configured paths to the GIM-II executive through the NSW Interactive Batch Specifier (IBS) and the CCN NSW Encapsulating Foreman. The services are configured around an existing data base belonging to the Air Logistics Command of the U. S. Air Force, so in their present form they should not be considered generalized public offerings.

Under NSW, most of the capabilities of GIM-II are available to the user, with the significant exceptions of those operations which use data sets or devices local to the host system. The NSW GIM-II user is constrained to operate with a predefined data base and with simple input and output files. In the interactive case, input and output are the users terminal and optional NSW files to receive the output of the NSW PRINT command. In the batch case, input and output are NSW files.

Access to a pool of backup tape reels is provided through the GIMDUMP and GIMRESTORE services; however, these are not intended for use by the average GIM-II user.

11.3 THE GIM INTERACTIVE SERVICE

The GIM service provides access from the NSW front end to the long-running instance of the GIM-II executive, through the CCN Encapsulating Foreman or WSCI.

11.3.1 Invoking GIM

When you type the USE command that invokes the GIM service, these events should occur:

1. Your terminal is connected to the CCN Encapsulator

2. If you are the first GIM user for a while, then the long-running instance of the GIM-II executive may not be active, and your session will pause briefly to activate it. If this happens, you will see

   ```
   The GIM system is being bootstrapped...
   Enter a carriage-return when asked for its password.
   ENTER PASSWORD:
   ```

   followed by an obscuring "blot". Do not attempt to provide a password -- just enter a carriage return. If the GIM executive is active to begin with, this step is skipped.

3. When the CCN Encapsulating Foreman is satisfied that the GIM-II executive is running, it issues the message

   ```
   *** UCLA-OAC NSW GIM SERVER ***
   ```

   and attempts to establish communication with it. If delays are encountered, you may see the message

   ```
   TRYING TO CONNECT TO GIM-II . . .
   ```

   A successful connection is indicated by the message

   ```
   GIM 4.2D CONNECTION = GIMZnnn
   ```

   An unsuccessful connection would be indicated by

   ```
   CANNOT CONNECT TO GIM-II
   ```

   and would skip to step 6.

4. When you see the GIM-II prompt ":", you begin your dialog with GIM-II. The first step should be to SIGNON to GIM using your assigned GIM-II signon parameters and password.

5. Eventually you issue the SIGNOFF command. You will see the message

   *** GIM CONNECTION CLOSED ***

   and your terminal will be returned by the encapsulator back to the Front End where you issued the USE command.

6. The Front End prompts you for another NSW command.

11.3.2 GIM Terminal-Handling Conventions

The NSW GIM service supports the following terminal protocol:

1. The normal GIM-II command prompt is ":", to which the user ought to respond with any of the valid GIM-II commands.

2. The interrupt-command prompt is "INT:", to which the user ought to respond with any of the valid GIM-II interrupt commands.

3. Status reporting or modification of a currently executing GIM-II is done through interrupt commands. These commands are entered in response to an interrupt prompt, which is itself produced by entering "attention" (control-C) while the command is executing.

4. Of particular interest is the DISPLAY ALL command. This command puts the terminal into an "automatic" mode; GIM does not prompt for terminal input, but only logs output directed to the terminal, as from other GIM terminals. To revert to normal control mode signal attention and respond to the "INT:" prompt with a "CT" command (return to control mode).

5. No form of statement terminator is used. Continuation lines may be entered by appending a "+" to every line but the last in the sequence. In particular "aaaaaa ++" specifies both a continuation line and the passing of a "+" as data.

6. All user input is translated to upper case.

7. Null lines are passed as such to GIM-II.

8. When the PRINT command is entered from the user's terminal, the data thus spun off is captured by the NSW Encapsulator. Before the session continues, the user will be asked to enter an NSW file name to which that data should be delivered.

11.4 THE BGIM SERVICE

BGIM is a batch service that executes an instance of the GIM-II executive program against the same data base that the GIM interactive service uses. BGIM uses an input file and an output file instead of the user's terminal. You will be prompted for NSW file names for those two files, and for a CPU time estimate. The default time estimate will probably be sufficient.

The input file will be converted to type 360-LIST. This means that the lines that you supply are significant only up to the carriage return. Unlike the GIM service, BGIM does expect statement terminators. All input records are concatenated up to the first occurrence of the reserved terminator "#".

The output file will be of type 360-PRINT. The standard batch-job system-output (SYSOUT) file is also produced.

Interception of the output from the PRINT verb into a separate NSW file is not available under BGIM.

11.5 THE GIMDUMP SERVICE

GIMDUMP is a batch service intended for use by personnel charged with maintaining the database used by GIM and BGIM. GIMDUMP defines a pool of ten magnetic tape reels, numbered 1 through 10. The user is prompted for a reel number and a name for a standard SYSOUT file. GIMDUMP dumps the GIM database onto the indicated reel. It is the user's responsibility to choose reels using a scheme that maintains some orderly backup system.

11.6 THE GIMRESTORE SERVICE

GIMRESTORE is a batch service intended for use by personnel charged with maintaining the database used by GIM and BGIM. It restores the GIM database from one of the tapes created by GIMDUMP. The user is prompted for a reel number and a name for a standard SYSOUT file.

11.7 EXAMPLES

```
NSW: USE ...GIM [confirm] !
  [command initiated]
  [command completed; 33GIM ready to use]
  [now talking to 33GIM (...gim)]
Welcome to UCLA/NSW GIM-II.
Enter GIM statements in response to the ":" prompt.
Enter GIM interrupt commands in response to the "INT:" prompt.
*** UCLA-OAC NSW GIM SERVER ***
GIM 4.2D CONNECTION = GIMZ098
: SIGNON ORG"A" OPER"5" DB"AFLCDB"
STATEMENT=  36982.000,  DATE=11/19/82,  TIME=13.6018,  USER=
ENTER PASSWORD:
SIGNON PROCESSED DATE=11/19/82,  TIME=13.6018
YOUR LAST SESSION ON THIS DATABASE WAS '11/11/82*11:00'.
 NO MESSAGES ON A          QUEUE.
COMPLETED 36982.000        CTIME=13.6038
: ROUTE *X
STATEMENT=  36983.000,  DATE=11/19/82,  TIME=13.6086,  USER=A
ROUTE LIST ESTABLISHED.
COMPLETED 36983.000        CTIME=13.6087
: /EX US
ORG     OPERATOR   UNIT ID    UNITS    DBNAME    SIGNON  TIME     STMT
                              099/100  AFLCDB            13:28       0
GSG     HELMS                 085/085  AFLCDB    13:31   13:34       0
A       5                     098/098  AFLCDB    13:36   13:36       0
PRINT4  P4          SPNOFF    054/054  AFLCDB    13:28               0
PRINT3  P3          SPNOFF    053/053  AFLCDB    13:28               0
PRINT2  P2          SPNOFF    052/052  AFLCDB    13:28               0
PRINT1  P1          SPNOFF    051/051  AFLCDB    13:28               0
PRINT0  P0          SPNOFF    050/050  AFLCDB    13:28               0:
PRINT

NSW file name:  GIMREPORT
  Full name of new file is
    $CCN.GIMREPORT  [now talking to 33GIM (...gim)]
new NSW file is $CCN.GIMREPORT
STATEMENT=  36985.000,  DATE=11/19/82,  TIME=13.6228,  USER=A
002 MESSAGES MOVED TO PRINT QUEUE FROM Q=5
COMPLETED 36985.000        CTIME=13.6247
: SIGNOFF
STATEMENT=  36986.000,  DATE=11/19/82,  TIME=13.7311,  USER=A
SIGNOFF ACKNOWLEDGED.  DATE=11/19/82,  TIME=13.7311
STATEMENTS PROCESSED=000004,  ELAPSED TIME= 0.1293
COMPLETED 36986.000        CTIME=13.7324
*** GIM CONNECTION CLOSED ***

  [Service termination initiated by 33GIM]
  Remote host closed connection
NSW:
  [35: Service termination completed; output ready; 33GIM closed]
NSW:
  [35: " Service termination " ]
```

```
Resources used by tool:
3   - CPU seconds
3   - Connect minutes
168   - I/O operations
The charges for this session are: 105



NSW: use ...bgim [confirm] !
  [command initiated]
  Beginning specification of job for batch tool
                PUBLIC.SERVICES.BGIM
  Time estimate in seconds (default 20):  [confirm] !
  Input file: bgim1.in [confirm] !
          Input file is CCN.BGIM1.IN
  Output file (default BGIM.OUTPUT):  [confirm] !
          Output file is CCN.BGIM.OUTPUT
  SYSOUT file (default BGIM.SYSOUT):  [confirm] !
  [command completed]
          Output file is CCN.BGIM.SYSOUT
  Job number is 50215
  Specification of batch job complete

NSW: show stat job 50215 [confirm] !
  [command initiated]
  [command completed]
  The status of job number 50215 for COMPASS + LUDLAM is: Sending
    Job processing begun
    Job allocated on batch host

NSW:
  The status of job number 50215 for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.BGIM1.IN ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 5673
    Execution complete per call to batch host
    File CCN.BGIM.SYSOUT delivered
    File CCN.BGIM.OUTPUT delivered
```

Section 12

SUPPORT FOR THE AP-1 SOFTWARE MAINTENANCE SYSTEM

12.1 FUNCTIONAL DESCRIPTION OF AP-1

The IBM AP-1 processor is the flight computer for the F-15 aircraft. The AP-1 Assembler, linker, subroutine package, and simulator are programs which support software development for that computer [44].

Only the Assembler and Linker are supported through NSW, and the ABSLIST function of the linker is somewhat restricted. Both services are batch services.

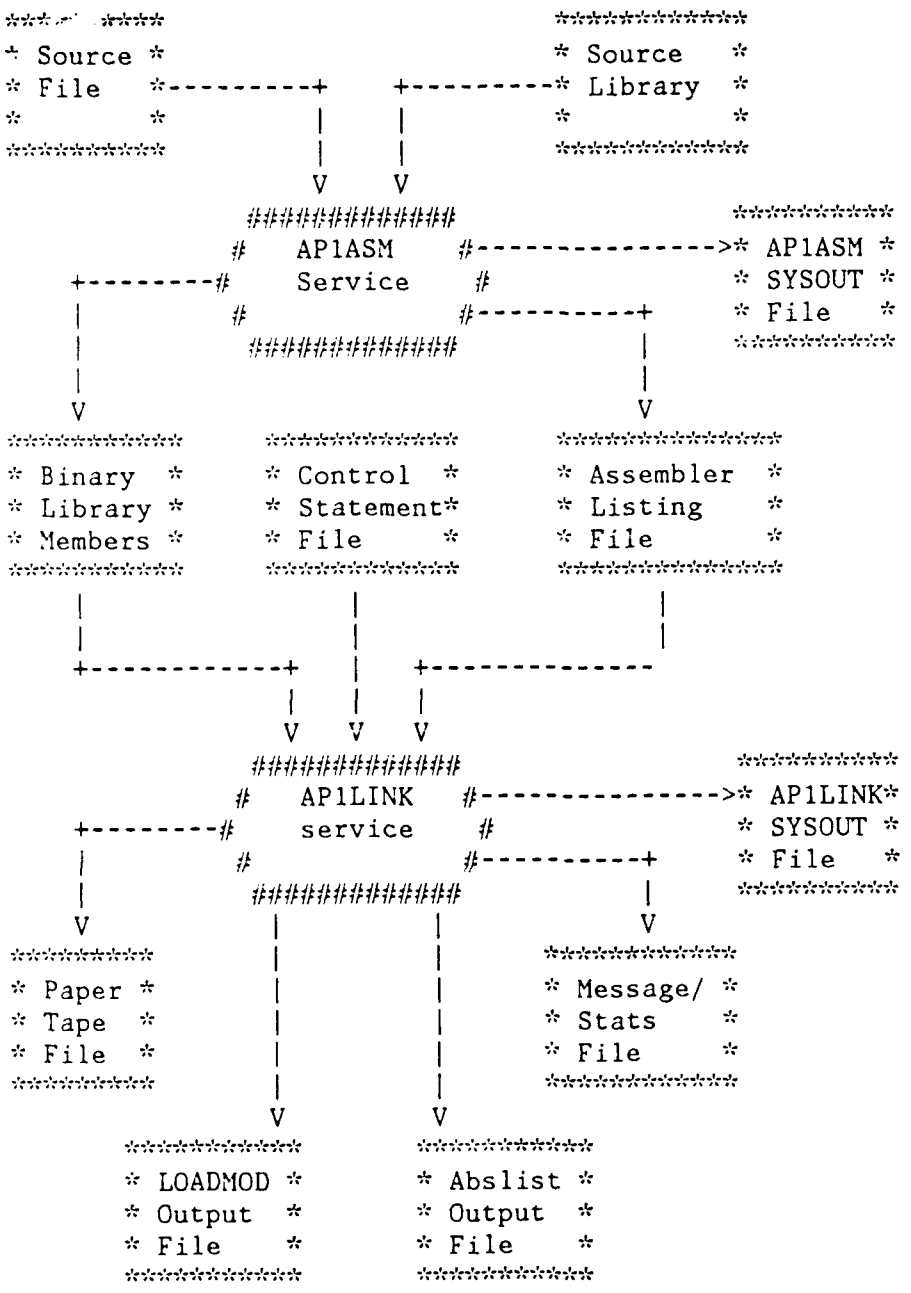


Figure 4: How the AP-1 Services Use Files

12.2 FILES USED BY THE AP-1 SERVICES

Figure 4 shows the relationships among the AP-1 services and the files that they use.

1. The AP-1 Source File

   This file consists of assembler-language statements on card images. Its preferred Global File Type (GFT) is 360-ASM-SOURCE. The records have optional sequence numbers in columns 73-80. You can prepare this data with any convenient editor, but if you do use sequence numbers, be sure that the file is delivered using a GFT that supports sequence numbers.

2. The AP-1 Source Library

   This card-image library consists of multiple members, each of which is formatted exactly like the AP-1 Source File. You create this library with the CREATEL service, and you place members into it with the PUTMEM service. If your application does not use library data, you do not need this file.

3. The Assembler Listing File

   AP1ASM produces a listing file of type 360-PRINT. This file may optionally be used as input to AP1LINK, when the ABSLIST option is used.

4. The Binary Library

   AP1ASM produces a binary output of type 360-OBJECT. If you wish to save this output, you must create an object library with the CREATEL service. This library must exist in order to use AP1LINK.

5. The SYSOUT Files

   AP1ASM and AP1LINK produce SYSOUT files, as do all CCN batch services.

6. The Control Statement File

   AP1LINK reads its control cards from this file. The data that you supply will be converted to type 360-CARDS.

7. The Message/Stats File

   AP1LINK produces its primary printer output on this file, which is of type 360-PRINT. This file does not include any printer output from the ABSLIST option.

8. The Paper Tape File

   AP1LINK uses this file to produce an image of the paper tape that is to be loaded into the AP-1 computer itself. Its GFT is 360-BINARY.

9. The LOADMOD File

   AP1LINK uses this file to produce an alternate form of the load data that is also written to the paper-tape file. This is the file that would be loaded into the AP1 Simulator, which is not available under NSW at this writing. Its GFT is 360-BINARY.

10. The Abslist Output File

    If the ABSLIST option is specified, AP1LINK produces a relocated copy of the Assembler Listing file on this file. It is also of type 360-PRINT.

12.3 THE AP1ASM SERVICE

AP1ASM is a batch service that assembles a single AP-1 routine. You are prompted for a CPU time estimate and for all the files that the assembler can use. You do not need to specify a source library or an object library if you do not wish the corresponding functions.

Because the listing file can vary drastically in size, the service asks you to supply a very rough estimate of the number of pages to allow for. This estimate helps the service make an intelligent disk allocation for the file.

The default time estimate of 20 seconds should be sufficient for an AP-1 assembly of moderate size.

At this time, there is no way to pass non-default parameters to the AP-1 Assembler.

12.4 THE AP1LINK SERVICE

AP1LINK is a batch service that combines the object outputs of various assemblies into a module that can be loaded into the AP-1 computer or the simulator. You are prompted for a CPU time estimate and for all the files that the linker can use. If you are not using ABSLIST you will not need to specify the two files that are concerned with ABSLIST processing.

Both the message file and the ABSLIST output file can vary drastically in size, so in each case, the service asks you to supply a very rough estimate of the number of pages to allow for. This estimate helps the service make intelligent disk allocations.

The default time estimate of 30 seconds should be sufficient for an average run.

The Assembler listing file that ABSLIST reads is normally the concatenation of the listing files produced by the assemblies that are being linked. Since NSW does not support concatenation at read time, you must have done the concatenation with an editor or similar tool before using AP1LINK. This is a restriction that will probably be lifted in a future NSW release.

At this time, there is no way to pass non-default parameters to the AP-1 Linkage Editor.

12.5 EXAMPLES

```
NSW:  use ...aplasm [confirm] !
  [command initiated]
  Beginning specification of job for batch tool
           PUBLIC.SERVICES.CCN.AP1ASM
  Time estimate in seconds (default 20): [confirm] !
  Source file:  ...mathsubs [confirm] !
          Input file is CCN.TESTA.MATHSUBS
  Will you be using a macro source library?
               (Yes or No): y [confirm] !
  Source library name: ...aplmacs [confirm] !
          Input file is CCN.TESTA.AP1MACS
  Assembler listing file (default AP1ASM.LISTING): [confirm] !
          Output file is CCN.TESTA.AP1ASM.LISTING
  Approximate printer pages (default 100): [confirm] !
  Will you save the object program?
               (Yes or No): y [confirm] !
  Library to receive object program: ...aplobj [confirm] !
          Input file is CCN.TESTA.AP1OBJ
  Object member name: mathsubs [confirm] !
  SYSOUT file (default AP1ASM.SYSOUT): [confirm] !
          Output file is CCN.TESTA.AP1ASM.SYSOUT
  [command completed]
  Job number is 50264
  Specification of batch job complete

NSW:
  The status of job number 50264
               for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.TESTA.MATHSUBS ready at batch host
    File CCN.TESTP.AP1MACS ready at batch host
    File CCN.TESTP.AP1OBJ ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 6270
    Execution complete per call from batch host
    File CCN.TESTP.AP1ASM.SYSOUT delivered into NSW
    File CCN.TESTP.AP1ASM.LISTING delivered into NSW

NSW:  use ...ap1link [confirm] !
  [command initiated]
  Beginning specification of job for batch tool
           PUBLIC.SERVICES.CCN.AP1LINK
  Time estimate in seconds (default 30): [confirm] !
  Control statement file: ...linksubs [confirm] !
          Input file is CCN.TESTA.LINKSUBS
  Msgs and stats file (default AP1LINK.MESSAGES): [confirm] !
          Output file is CCN.TESTA.AP1LINK.MESSAGES
  Est. pages in msgs and stats file (default 100): [confirm] !
  Object program library: ...aplobj [confirm] !
          Input file is CCN.TESTA.AP1OBJ
  Paper tape load file (default AP1LINK.PTAPE): [confirm] !
```

```
          Output file is CCN.TESTA.AP1LINK.PTAPE
  LOADMOD output file (default AP1LINK.LOADMOD): [confirm] !
          Output file is CCN.TESTA.AP1LINK.LOADMOD
  Are you invoking the ABSLIST option?
               (Yes or No): n [confirm] !
  SYSOUT file (default AP1LINK.SYSOUT): [confirm] !
          Output file is CCN.TESTA.AP1LINK.SYSOUT
  [command completed]
  Job number is 50265
  Specification of batch job complete

NSW:
  The status of job number 50265
               for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.TESTA.LINKSUBS ready at batch host
    File CCN.TESTP.AP1OBJ ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 6289
    Execution complete per call from batch host
    File CCN.TESTP.AP1LINK.SYSOUT delivered into NSW
    File CCN.TESTP.AP1LINK.LOADMOD delivered into NSW
    File CCN.TESTP.AP1LINK.PTAPE delivered into NSW
    File CCN.TESTP.AP1LINK.MESSAGES delivered into NSW

NSW:
```

Section 13

SUPPORT FOR THE B52 OFFLINE AVIONICS SYSTEM

13.1 FUNCTIONAL DESCRIPTION OF THE B52OAS

The B52OAS support package is a collection of services implementing the various functions of the B52OAS database-manager, assembler, compiler, linker and simulator [1]. These are all batch services.

As configured under NSW, the B52OAS services include both programs and pre-configured data bases.

The following data bases are provided:

```
FCPDB     - source B52OAS database ("IF" database)
GMCPDB    - supplemental library B52OAS database ("SL" database)
CHANGEDB - change database ("OF" database)
APCLIB    - scratch database
TEMP      - temporary database to pass results between
            the service calls, i.e. assembler to simulator.
```

The following Services are provided:

```
B52APCJ - J3B compiler, AP/101C assembler output
B52IBMJ - J3B compiler, IBM assembler output
B52SAMP - Cross reference processor
B52APCA - AP/101C assembler
B52SIMU - AP/101C simulator
B52DBMS - DBMS (database management processor)
```

Each service requires an input file (converted to type 360-CARDS), and an output file and a SYSOUT file (each of type 360-PRINT). The data-base files are built into the package, and are not prompted for. Each service will also prompt for a CPU time estimate. The default values have been chosen to suffice in typical cases.

13.2 EXAMPLE

```
NSW: use ...b52apcj [confirm] !
  [command initiated]
  Beginning specification of job for batch tool PUBLIC.SERVICES.B52APCJ
  Time estimate in minutes (default 1):  [confirm] !
  Input file: ...apcj.in [confirm] !
          Input file is CCN.B52.APCJ.IN
  Output file (default B52APCJ.OUTPUT):  [confirm] !
          Output file is CCN.B52APCJ.OUTPUT
  SYSOUT file (default B52APCJ.SYSOUT):  [confirm] !
  [command completed]
          Output file is CCN.B52APCJ.SYSOUT
  Job number is 50214
  Specification of batch job complete

NSW: show stat job 50214 [confirm] !
  [command initiated]
  [command completed]
  The status of job number 50214 for COMPASS + LUDLAM is: Submitting
    Job processing begun
    Job allocated on batch host
    File CCN.B52.APCJ.IN ready at batch host

  The status of job number 50214 for COMPASS + LUDLAM is: Completed
    Job processing begun
    Job allocated on batch host
    File CCN.B52.APCJ.IN ready at batch host
    SYSIN file ready at batch host
    STARTJOB accepted for batch host job name 5495
    Execution complete per call to batch host
    File CCN.B52APCJ.SYSOUT delivered
    File CCN.B52APCJ.OUTPUT delivered
```

REFERENCES

1 The Boeing Corporation, *Control Program Operational Procedures: B52OAS Support Computer Program - Milestone 7.* Document no. D675-10107-701.

2 IBM Corporation, *OS/VS2 MVS JCL.* Order no. GC28-0692.

3 IBM Corporation, *OS/VS2 TSO Command Language Reference.* Order no. GC28-0646.

4 IBM Corporation, *General Purpose Simulation System V Introductory User's Manual.* IBM Form SH20-0860

5 IBM Corporation, *General Purpose Simulation System V User's Manual.* IBM Form SH20-0851

6 IBM Corporation, *OS/VS2 TSO Command Language Reference.* IBM form no. GC28-0646.

7 IBM Corporation, *OS/VS Linkage Editor and Loader.* IBM form no. GC26-3813.

8 IBM Corporation, *OS/VS Linkage Editor and Loader Messages.* IBM form no. GC38-1007.

9 IBM Corporation, *IBM SYSTEM/360 and SYSTEM/370 FORTRAN IV Language.* IBM form no. GC28-6515.

10 IBM Corporation, *IBM SYSTEM/360 Operating System FORTRAN IV (G and H) Programmer's Guide.* IBM form no. GC28-6817.

11 IBM Corporation, *OS PL/I Checkout and Optimizing Compilers: Language Reference Manual.* IBM form no. GC33-0009.

12 IBM Corporation, *OS PL/I Optimizing Compiler: Programmer's Guide.* IBM form no. GC33-0006.

13 IBM Corporation, *OS PL/I Optimizing Compiler: TSO User's Guide.* IBM form no. GC33-0029.

14 IBM Corporation, *OS PL/I Optimizing Compiler: Messages.* IBM form no. GC33-0027.

15 IBM Corporation, *IBM SYSTEM/370 Principles of Operation.* IBM form no. GA22-7000.

16 IBM Corporation, *OS/VS-DOS/VSE-VM/370 Assembler Language*. IBM form no. GC33-4010.

17 IBM Corporation, *OS Assembler H Language*. IBM form no. GC26-3771.

18 IBM Corporation, *OS Assembler H Programmer's Guide*. IBM form no. GC33-3759.

19 IBM Corporation, *OS Assembler H Messages*. IBM form no. GC33-3770.

20 IBM Corporation, *OS/VS2 MVS Supervisor Services and Macro Instructions*. IBM form no. GC28-0683.

21 IBM Corporation, *OS/VS2 MVS Data Management Services Guide*. IBM form no. GC26-3875.

22 IBM Corporation, *OS/VS2 MVS Data Management Macro Instructions*. IBM form no. GC26-3873.

23 IBM Corporation, *OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor*. IBM form no. GC28-0648.

24 IBM Corporation, *ECAP/360 - Electronic Circuit Analysis Program*. IBM S/360 General Program Library Form 360D-16.4.001.

25 IBM Corporation, *IBM 1620 Electronic Circuit Analysis Program ECAP (1620--EE-02X): User's Manual*. IBM Form H20-0170-1.

26 Lind, Henrik O., *NSW User Reference Manual*. BBN, May, 1982.

27 Ludlam, N., *Using the UCLA Interim Library Management Tool Kit*. Document UCNSW-101, Office of Academic Computing, UCLA, July 18, 1979.

28 Ludlam, N., *Using the UCLA Native Language Processing Tool Kits*. Document UCNSW-102, Office of Academic Computing, UCLA, July 18, 1979.

29 Ludlam, N., *Using the UCLA PL/I Tool Kit*. Document UCNSW-103, Office of Academic Computing, UCLA, July 18, 1979.

30 Ludlam, N., *Using the Display Tool*. Document UCNSW-106, Office of Academic Computing, UCLA, March 1, 1980.

31 Ludlam, N., *Using the TSOEDIT Tool*. Document UCNSW-107, Office of Academic Computing, UCLA, July March 1, 1980.

32 Ludlam, N., *Installing the UCLA PL/I Tool Kit*. Document UCNSW-211, Office of Academic Computing, UCLA, November 6, 1979.

33 Ludlam, N., *Installing the UCLA ILM Tool Kit*. Document UCNSW-212, Office of Academic Computing, UCLA, November 6, 1979.

34 NATS Project, *Eigenoystem Subroutine Package (EISPACK)*, Subroutines E2F269 to E2F298 and E2F220 to E2F223.

35 Nie, Norman H., et al. *Statistical Package for the Social Sciences, 2nd ed.* McGraw-Hill, Inc., 1975.

36 TRW Systems Group, *GIM-II Brochure.* Document 6760-W500-RU-00.

37 TRW Systems Group, *GIM-II Executive Summary.* Document 6760-W510-RU-00.

38 TRW Systems Group, *GIM-II Reference Manual.* Document 6760-W520-RU-00.

39 TRW Systems Group, *GIM-II Acceptance Test Plans.* Document 6760-W530-RU-00.

40 TRW Systems Group, *GIM-II Procedure Oriented Language.* Document 6760-W540-RU-00.

41 TRW Systems Group, *GIM-II Data Base Operation Manual.* Document 6760-W560-RU-00.

42 TRW Systems Group, *GIM-II Release 4.2D update notes.*

43 UCLA Office of Academic Computing, *Basic JCL.* Document B003.

44 McDonnell Douglas Corporation, *AP-1 Computer Software Systems Manual.* Document 6007340A, September 15, 1972.